Don Thomasson

AMSTRAD

Desensamblado de la ROM y Mapa de Memoria

CPC 464-664-6128

AMSTRAD

DESENSAMBLADO DE LA ROM Y MAPA DE MEMORIA

AMSTRAD
Desensamblado de la ROM y mapa de memoria

Don Thomasson



MICROINFORMATICA

Título de la obra original:
AMSTRAD CPC464 WHOLE MEMORY GUIDE

Traducción y adaptación: Rafael Sarmiento de Sotomayor
Diseño de colección: Antonio Lax
Diseño de cubierta: Narcís Fernández

Primera edición, 1986
Primera reimpresión, 1987

Edición publicada por acuerdo con
Melbourne House (Publishers) Ltd.,
Londres

Villafranca, 22. 28028 Madrid
Depósito legal: M. 31.689-1987
ISBN: 84-7614-103-3
Printed in Spain
Imprime: Anzos, S. A. - Fuenlabrada (Madrid)

Indice

Introducción 9

1. **Disposición general del sistema** 13

El mapa de memoria 13
El mapa de E/S 15
Periféricos externos 16
Los estados del sistema 16
Las entradas del bloque de saltos 17
Sumario 18
Convenios 18

2. **Las rutinas de la RAM** 21

El área RST 22
Rutinas RAM del bloque de saltos 25
Area de extensiones RST 28
Comentarios 33

3. **El control de la máquina** 35

Reinicialización principal 35
Rutinas de impresora 39
Otras rutinas MC 41

4. El núcleo 45

El controlador de interrupciones 46
El sistema de sucesos 48
Sucesos síncronos 53
Comentarios 56
Otras rutinas del núcleo 56
Area de datos del núcleo 57

5. Sistema de pantalla 59

La RAM de pantalla 59
Cauces y ventanas 61
Parámetros 62
El espacio de trabajo 62
Compresión de pantalla 62
Gráficos 63
VDU de texto 63

6. El control de pantalla 67

Control del modo 68
Tabla de máscaras 69
Direcciones 69
Tintas y parpadeo de colores 72
El sistema de parpadeo 74
Rutinas llamadas por PNT borra pantalla 75
Rutina de sucesos 75
Rutinas generales 76
Comentarios 81

7. La unidad de video para texto 85

Control de pantalla y cursor 86
Color 90
Ventanas 91
Cauces 92
Datos de las matrices 93
Salida de textos 95
Tabla de carácter de control 98
Otras rutinas de texto 99
Comentarios 100

8. La unidad de video para gráficos 103
Actualización 104
Comprobando valores 105
Funciones principales 106
Comentarios 109

9. El monitor de teclado 111

Rutinas de teclado 113
Rutinas de entrada 114
Cadenas de teclas 115
Tablas de teclas y códigos 118
Acción repetitiva 120
Funciones de ruptura 121
Sumario 122
Mapas de bits 123
Tablas tecla/código 123
Espacio de trabajo del monitor de teclado 124

10. El monitor de cassettte 127

Mensajes 129
Las rutinas 130
Llamadas diversas 135
Tipos de ficheros 136
Comentarios 137
Espacio de trabajo para el cassette 137

11. El monitor de sonido 141

Llamadas del sistema 143
Comentarios 147
Espacio de trabajo del monitor de sonido 148
Formato del *buffer* de un canal 148

12. Las ROM externas 151

Ordenes 152
Rutinas 154

13. Soporte del BASIC 159

La coma flotante 159
Localización de las entradas 161
Manejando llamadas para matemáticas 164

14. El intérprete de BASIC 167

Palabras reservadas en orden de clave 168

Apéndice A. Programas de soporte 171
Apéndice B. Indice por direcciones 183
Apéndice C. Mapa de memoria 189
Apéndice D. Rutinas específicas del 6128 191
Indice alfabético 251

Introducción

Los Amstrad CPC464 y CPC6128 son aparatos fascinantes en muchos sentidos, pero pueden resultar irritantes cuando se desconocen algunos detalles esenciales sobre funciones internas. Incluso contando con un juego completo de documentación oficial, que podría ocupar varios volúmenes, aún quedarían algunos puntos oscuros.

El sistema operativo, por ejemplo, tiene más de 250 puntos de entrada, cada uno relacionado con una función específica, pero, de estas entradas, unas 50 no están definidas oficialmente, porque se consideran esencialmente como extensiones del intérprete de BASIC. Entre las demás entradas, hay algunas definidas de tal forma que su función no queda del todo clara inmediatamente. Se necesita completar la visión general contando con las explicaciones oficiales imprescindibles. Este libro intenta cubrir esa necesidad.

Un análisis detallado del sistema operativo sería muy largo y tedioso, e incluso podría no responder a todas las cuestiones que se plantean. Así que la idea es hacer un análisis de las funciones más importantes, aludiendo al resto con descripciones cortas.

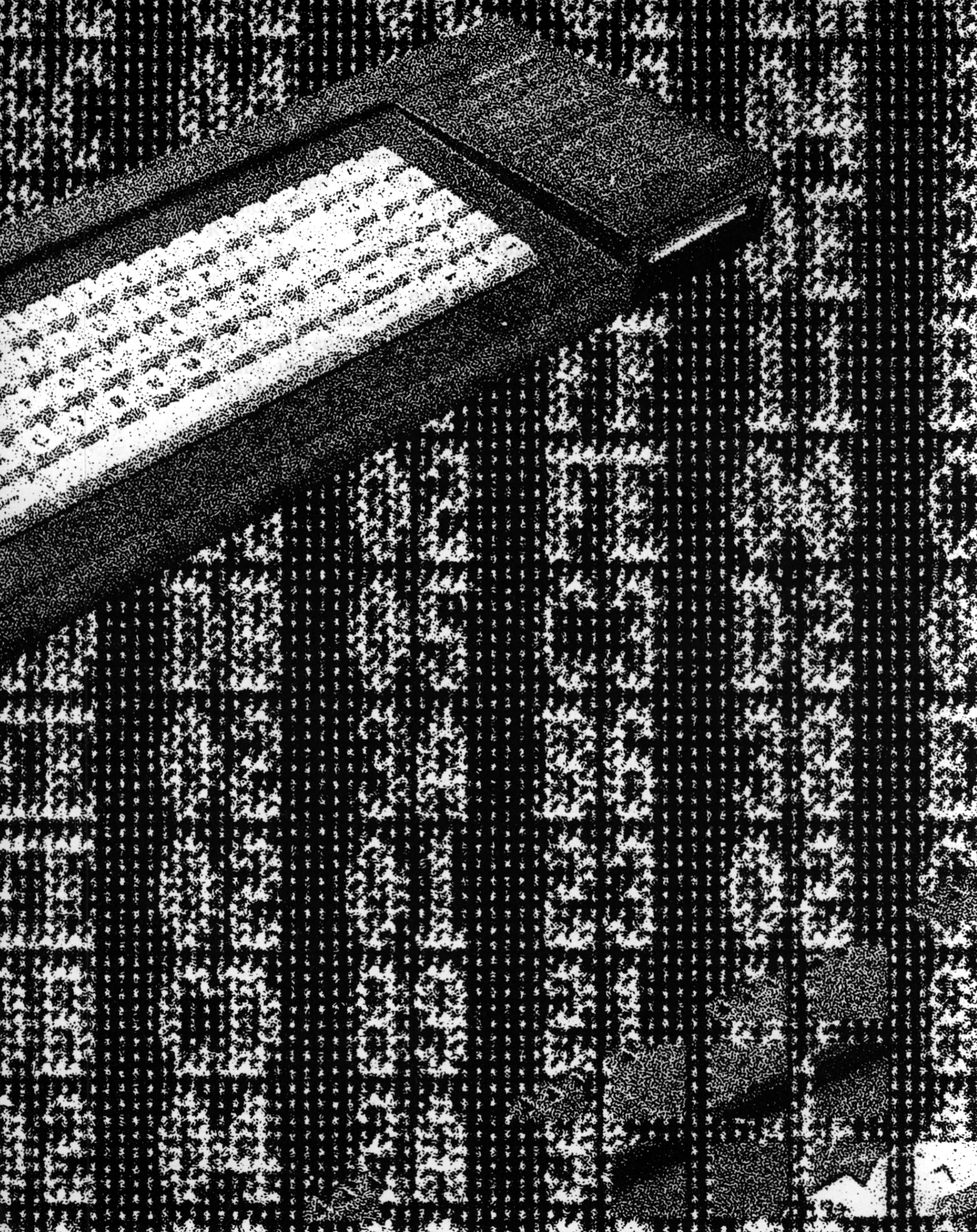
Se presume que el lector tiene conocimientos de código máquina. Incluir aquí una sección completa sobre la programación del Z-80 no dejaría apenas espacio para otras cosas. Para aquellos que necesiten esta ayuda, es de recomendar el conocido libro de Rodnay Zaks *Programación*

*del Z-80**. Sin embargo, un estudio de las distintas rutinas del sistema operativo, en relación con las descripciones que se ofrecen a continuación, resultará muy instructivo y lleno de datos útiles, incluso para el principiante.

Una dificultad clara es que algunos desensambladores sólo tienen acceso al código que está en la ROM. Un programa que se presenta en el apéndice propone una solución, puesto que funciona desde el código de la ROM, mientras que otro programa también del apéndice ofrece métodos adecuados para llamar a las distintas rutinas funcionales y comprobar su acción.

El libro se basa en la versión 1.0 de las ROM, pero los comentarios podrán aplicarse sin problema a otras versiones, trabajando desde los puntos de entrada al bloque de saltos, que permanecerá en las direcciones dadas, incluso si acceden por distintos puntos de entrada al código de la ROM.

* Zaks, Rodnay: *Programación del Z-80*, Anaya Multimedia, Madrid, 1985.



1 Disposición general del sistema

En términos generales, el CPC464 es un sistema típico basado en el microprocesador Z-80, pero con una ordenación de los dispositivos periféricos poco usual en cuanto a economía. Mediante una explotación al máximo de las capacidades de estos dispositivos, se ha obtenido un nivel de rendimiento mayor de lo que en principio puede sugerir el *chip*. Una consecuencia de esto es la complejidad que presenta el sistema operativo, hecho este que se compensa al compararlo con las facilidades ofrecidas al usuario para el acceso a las diferentes funciones. La palabra "comparación" es necesaria, porque es preciso un cierto conocimiento del código máquina, lo cual puede ser una dificultad para algunos usuarios, pero, una vez que se entiende el tema del lenguaje máquina, se abre un amplio abanico de posibilidades.

Entre otras genialidades, es especialmente notable la forma en que han sido dispuestos un mínimo de 96K de memoria en un mapa de memoria de sólo 64K. Este será el primer aspecto del sistema que se va a estudiar.

El mapa de memoria

La totalidad de los 64 Kbytes de la memoria son ocupados por la RAM, en la que se puede escribir directamente. Y esto tiene sentido, ya que no sería lógico intentar escribir sobre la ROM. Las lecturas de

direcciones en la zona media de la memoria accederán siempre a la RAM, ya que no hay ROM en esta área. En las direcciones correspondientes a la cuarta parte inferior y superior de la memoria, tanto RAM como ROM, están presentes y es posible leer en cualquiera de ellas a voluntad. Una orden PEEK del BASIC accederá siempre a la RAM, pero es necesario un bit especial del código máquina para obtener los contenidos de la ROM.

La disposición de la memoria es complicada por el hecho de que la cuarta parte superior de la RAM se utiliza como memoria de pantalla y debe ser de acceso inmediato, a intervalos regulares, para que los datos se transfieran a la pantalla. Para este propósito, se leen dos bytes cada microsegundo.

El procesador se mantiene en estado de espera mientras se transfieren los pares de bytes. La transferencia se realiza directamente desde la memoria hasta la matriz de puertas de video *(Video Gate Array)*, mediante direcciones generadas por el *chip* controlador CRT (controlador del tubo de rayos catódicos). Esto significa que el procesador principal solamente puede acceder a memoria una vez por microsegundo y, aunque su reloj oscila a 4 MHz, la velocidad real de procesado está ligeramente reducida. Este es un punto a tener en cuenta a la hora de calcular los tiempos de ejecución.

La matriz de puertas de video gobierna la conmutación entre RAM y ROM para este propósito, por lo que es natural que también se utilice para controlar la selección de ROM. Las instrucciones para la conmutación entre RAM y ROM vienen dadas por los bits 2 y 3 producidos por la puerta 7FXX. Un 1 desactiva y un 0 activa, refiriéndose el bit 2 a la ROM inferior y el bit 3 a la ROM superior. Incidentalmente, existe un único componente ROM, que se divide en dos bloques de 16K, tan separados como pueda permitirlo el sistema.

Al igual que en cualquier sistema que funciona como banco conmutado de memoria, el problema clave es la necesidad de saltar y conmutar bancos simultáneamente, o al menos hacerlo aparentemente. Los CPC464 y CPC6128 consiguen esto mediante rutinas almacenadas en la zona media de la RAM. Estas son siempre accesibles, cualquiera que sea el estado seleccionado de la ROM. Con una sencilla conmutación entre ROM y RAM, estas rutinas permiten seleccionar ROM superiores alternativas, extendiendo la memoria disponible más allá todavía. En el límite, es posible direccionar un total de 4.128 Kbytes de memoria nominalmente, pero pocos sistemas parecen aproximarse a este último extremo.

Las complicaciones del sistema de memoria se pueden evitar introduciendo el código máquina en la zona central del mapa de memoria, que únicamente contiene RAM, pero esto no siempre es factible ni esencial.

El mapa de E/S

La selección de los canales periféricos viene determinada, en gran parte, poniendo a nivel bajo uno de los bits del byte alto de la dirección de 16 bits de E/S. Esto significa que no se puede utilizar las viejas instrucciones de la forma IN A,(N) y OUT (N),A, ya que toman el byte alto del contenido del registro A. Las instrucciones obligatorias son las que fijan las direcciones de E/S, a partir del contenido del registro BC, y hay límites estrictos que restringen los contenidos del registro B, ya que sólo uno de los seis bits superiores puede estar a nivel bajo para toda dirección dada. (Poner a cero más de uno de estos bits en una instrucción de entrada provocará daños físicos en el aparato, ya que dos fuentes de datos pelearán por apoderarse del *bus*, hasta que una de ellas se estropee. Por otra parte, es muy poco frecuente enviar la misma salida, a un mismo tiempo, a dos puertas diferentes.)

Las direcciones de E/S se pueden resumir de la siguiente manera:

- Si el bit 15 de la dirección está a nivel bajo, queda seleccionada la matriz de puertas de video. Esta puerta es únicamente para salidas. La dirección debe ser 7FXX.
- Si el bit 14 de la dirección está a nivel bajo, el controlador CRT queda seleccionado. Los bits A8 y A9 de la dirección son utilizados para seleccionar uno de los cuatro posibles modos de transferencia:

BCXX	Salida al registro elegido
BDXX	Salida de datos
BEXX	Registro de estado de entrada
BFXX	Entrada de datos

- Si el bit 13 de la dirección está a nivel bajo, los datos de salida son de la ROM seleccionada. La dirección deberá ser DFXX.
- Si el bit 12 de la dirección está a nivel bajo, queda seleccionado el canal de la impresora que únicamente es de salida de datos. La dirección debe ser EFXX.
- Si el bit 11 de la dirección está a nivel bajo, el interfaz de periféricos en paralelo (PPI) queda seleccionado. Aquí nuevamente los bits A8 y A9 se emplean para distinguir uno de cuatro posibles subcanales:

F4XX	Puerto A (E/S)
F5XX	Puerto B (E/S)
F6XX	Puerto C (E/S)
F7XX	Control (sólo salidas)

- Si el bit 10 de la dirección está a nivel bajo, se selecciona un canal para expansiones. En este caso, los bits A5-A7 tienen un significado especial:

A5 a 0 selecciona un canal de comunicación
A6 a 0 selecciona una función reservada
A7 a 0 selecciona el sistema de disco

- La dirección F8FF es una reinicialización general para los canales de expansión.

Todas estas localizaciones limitan la gama de direcciones de que dispone el usuario para las funciones de E/S que pueda requerir. El rango de direcciones para el usuario es:

F8E0-F8FE ; F9E0-F9FF ; FAE0-FAFF ; FBE0-FBFF

Periféricos externos

Los dispositivos antes mencionados son los "periféricos internos", que son directamente accesibles desde el procesador principal. El resto de circuitos, clasificados como "periféricos externos", únicamente son accesibles desde los periféricos internos. Se incluyen en este grupo de externos: el generador programable de sonido o PSG, accesible desde la PPI; el teclado, accesible desde la PPI y el generador de sonido; el sistema de cassette, accesible desde la PPI; y, finalmente, el altavoz, que es gobernado por el generador de sonido.

Para ampliar detalles sobre el *hardware* del sistema, puedes consultar el libro *Programación avanzada del Amstrad**, que ofrece información adicional sobre los códigos y acciones de estos dispositivos.

Los estados del sistema

Al encender el aparato se ejecuta un cierto número de procesos de inicialización, pasando entonces el control a la ROM 0 superior. Si no existe dicha ROM superior, el intérprete interno de BASIC se hace cargo del control como programa primario.

Una vez que el programa primario está en funcionamiento, permanecerá así hasta que se ejecute un retorno al nivel de entrada, lo cual ocurrirá cuando haya finalizado la inicialización completa y la ROM 0 sea llamada de nuevo para tomar el control. Sin embargo, el programa primario puede llamar a programas secundarios para que el asistan y éstos, a su vez, pueden recurrir a otros programas. Por ello se dice que —nominalmente— tenemos un programa primario en funcionamiento al conectar el ordenador, pero pueden existir varios niveles secundarios.

Una ROM distinta de la 0, o un programa en RAM, se pueden

* Thomasson, Don: *Programación avanzada del Amstrad (Descripción de la ROM. Rutinas y parámetros)*, Anaya Multimedia, Madrid, 1985.

seleccionar como programa primario. Esto puede hacerse mediante una orden RUN " " que leerá un programa en código máquina, que tiene definida una dirección de comienzo concreta, o por una rutina en código máquina. Puede ser más conveniente dejar al intérprete de BASIC a cargo de todo y ejecutar un programa llamado desde BASIC mediante CALL, como si éste fuera el auténtico programa primario. Esto tiene la ventaja de que no es inevitable una reinicialización completa cuando se devuelve el control al nivel de entrada. Es decir, un programa en RAM, que toma el carácter de primario, actuará como un sistema operativo construido por nosotros mismos, evitando así una reinicialización del aparato cuando por cualquier causa el programa primario vuelve a ejecutarse. Esto último no es posible hacerlo con el intérprete de BASIC.

El empleo del BASIC en esta forma ofrece otras ventajas. El valor de HIMEM puede ser examinado y ajustado de manera sencilla, poniéndolo por debajo del área donde reside el código máquina. Además se pueden inicializar otras variables del sistema. El programa en BASIC ocupa cierta cantidad de RAM, concretamente desde 0170 hacia arriba. En realidad, el espacio no utilizado es insignificante en relación al tamaño de la memoria disponible.

Un punto a observar es que, si se añaden sistemas de expansión tales como una unidad de disco, un facilitador de palabras, o el ensamblador MAXAM en su formato de ROM, entonces HIMEM se reduce, ya que las expansiones precisan de un cierto espacio de trabajo para su funcionamiento. Algunos programas comerciales son incompatibles con una unidad de disco, porque se apropian del espacio de trabajo del sistema de disco. Es decir, que, protegidos o sin proteger, no hay forma de transferirlos a un disco.

Como referencia en circunstancias normales, el valor de HIMEM es AB7F, pero queda rebajada a A67B con una unidad de disco conectada y puede ser mucho más bajo aún con el AMSDOS.

La advertencia oficial es que los programas en código máquina deberían ser relocalizables, pero esto no siempre es asequible. Se ha observado, sin embargo, que es posible introducir pequeñas rutinas en la zona BF00, ya que en esta área sobreviven a una reinicialización, lo cual es siempre de utilidad ...

Las entradas* del bloque de saltos

El área de la RAM comprendida entre BB00 y BDC9 contiene instrucciones de acceso a las principales entradas del sistema operativo. Se emplean saltos especiales para seleccionar automáticamente la ROM re-

* *N. del T.:* Se entiende por entrada un conjunto de instrucciones o datos que utiliza el programa principal como ayuda en la ejecución, no como subrutina. Un ejemplo de entrada son los datos que acompañan a una sentencia DATA.

querida. Las entradas que comienzan con &CF hacen esto y nada más, pero las entradas que comienzan con &EF, además, ihiben la ROM relevante al regreso de la rutina (véase el apartado "El área RST").

Las entradas del bloque de saltos son accesibles mediante llamadas a subrutina (CALL); la dirección de retorno queda almacenada y disponible en la pila.

Desde BDCD hasta BDF3 las entradas son simples saltos y comienzan con &C3. Estas son sólo "indirecciones" que no activan la ROM apropiada y que sólo se pueden llamar una vez que se sabe que la ROM ya está seleccionada y activada.

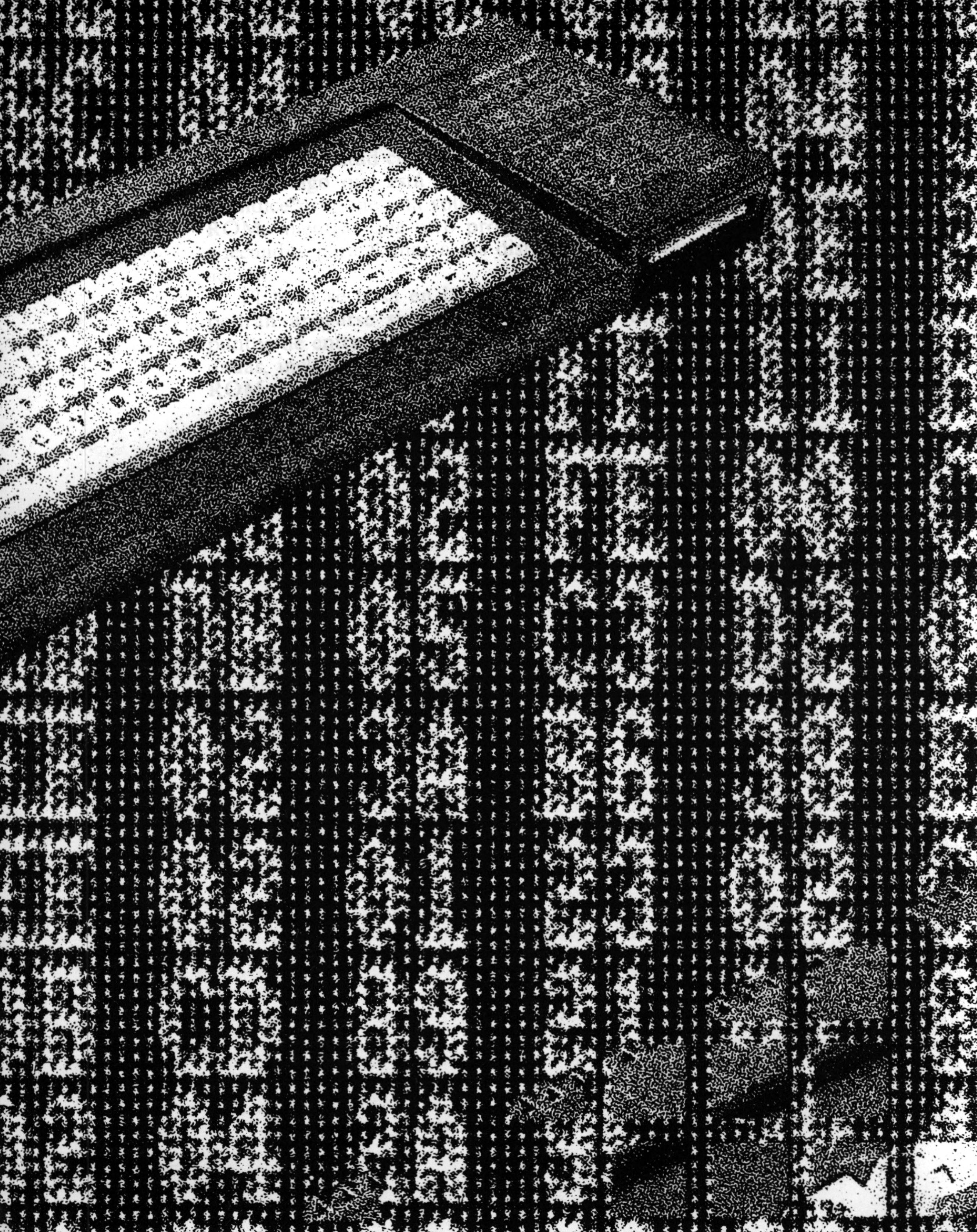
La intención es que las direcciones del bloque de saltos no se modifiquen de forma que puedan acceder a los diferentes puntos de entradas con diferentes versiones del sistema. Sin embargo, para facilitar la referencia, cada descripción de las rutinas viene encabezada por su dirección en el bloque de saltos y el destino asociado en el sistema operativo. Este último cambia con la versión del sistema, como en el CPC664 y CPC6128, y los nuevos puntos de entrada se deben determinar examinando las instrucciones del bloque de saltos.

Sumario

Este rápido recorrido por las características del sistema tendría que servir como útil introducción al sistema. Ahora vamos a comenzar a estudiarlo más detalladamente, empezando con las rutinas contenidas en la RAM.

Convenios

Los números con dos dígitos hexadecimales estarán precedidos por "&"; los números con cuatro dígitos hexadecimales, no. Los paréntesis alrededor de un número de cuatro dígitos indican el contenido de la posición identificada por el número. Donde los paréntesis contengan una gama de números, por ejemplo, en la forma (00FA/D), se entenderá el contenido de las posiciones especificadas por esos números. En este caso, el contenido de 00FA hasta 00FD.



2
Las rutinas de la RAM

Durante la inicialización se actualizan dos áreas de la RAM con una pequeña copia de la ROM. El resultado es un código máquina que se puede ejecutar, tanto si las ROM están activadas como si no, siendo esto tan sólo una parte de la historia.

El área 0000-003F se carga con las correspondientes localizaciones de la ROM. Esta es la denominada "Area RST", que contiene un cierto número de puntos de entrada especiales que necesitan ser efectivos en todo momento. Un interesante aspecto sobre la cuestión es que, al encender el ordenador, el procesador direcciona la posición 0000, pero en ese momento la RAM todavía no está actualizada con los valores apropiados y es a la ROM baja adonde acude realmente el procesador principal. Todo esto se asegura por la inicialización *hardware* de la matriz de puertas de video.

Algunos puntos del área RST pueden ser accesibles por las instrucciones RST del Z-80, pero el significado de esto ha sido modificado en el sistema CPC, haciendo que estos puntos actúen como rutinas a las que se accede con CALL desde la otra área de rutinas de la RAM, que se encuentra entre B900 y BAE8. Esta área sirve para un cierto número de propósitos:

B900-B920	Contiene un bloque de saltos para acceso a rutinas de la zona BA4A-BAB1.
B921-B938	Contiene NC PRIORIDAD (véase "Rutinas de sucesos").

B939-B97B	Contiene el manipulador principal de interrupciones.
B97C-BA49	Contiene las entradas de las rutinas implementadas en el área RST.
BA4A-BAB1	Contiene las rutinas de control de la ROM y de copia.
BAB2-BAE8	Contiene rutinas de lectura de la RAM.

Para simplificar la explicación, la acción de las rutinas en B97C-BAE8 se describirá primero en términos de su función y después se examinará con detalle el código, para aquellos que quieran saber más sobre su funcionamiento.

El área RST

Las instrucciones RST del Z-80 son de la forma &C7+X, y se refieren a una subrutina llamada en la dirección X. La dirección de retorno queda almacenada en la pila. X puede ser 0, 8, &10, &18, &20, &28, &30 o &38.

Los sistemas CPC464 y CPC6128 amplían el significado de estas instrucciones, haciéndolas rutinas de acceso en la RAM, lo cual modifica su efecto considerablemente.

RST00 (código &C7): entrada localizada en 0000 y, como en el encendido inicial, realizada una inicialización completa. El código es inmediatamente colocado por la matriz de puertas de video con una salida de &89 en 7FXX; después hay un salto a 0580 para finalizar el resto de la inicialización (véase capítulo siguiente).

RST08 (código &CF): localizada en 0008, donde se efectúa un salto a B982 de las rutinas de RAM. Los dos bytes que acompañan al código &CF son interpretados como una palabra de 16 bits de la siguiente forma:

Bits 0-13	Una dirección en la gama 0000-3FFF.
Bit 14	0 para activar la ROM inferior, 1 para inhibirla.
Bit 15	0 para activar la ROM superior, 1 para inhibirla.

Se pone la condición específica para la ROM y se realiza un salto a la dirección dada. Esta función, denominada INF SALTO, es una de las armas secretas que hacen practicable la conmutación de memorias del sistema, ya que el salto y el cambio de ROM aparentan ocurrir simultáneamente.

La entrada en 000B ejecuta un salto a B97C que está entre las rutinas de la RAM superior. Esta es la denominada SALTO HL, que es similar a INF SALTO, excepto que la palabra 16 bits está contenida en el registro HL.

La entrada en 000E ejecuta un salto a la dirección definida por el contenido del registro BC. Se conoce como SALTO BC, que simula la función de JP(BC).

Ninguna de estas dos entradas es accesible por una instrucción RST, pero sí pueden ser accesibles de la forma habitual con un salto o una llamada CALL.

RST10 (&D7): localizada en 0010, donde se realiza un salto a BA16 en las rutinas de la RAM superior. Esta implementa la función LLAMADA LATERAL. Los dos bytes que siguen al código &D7 se leen como una palabra de 16 bits y se interpretan de esta forma:

Bits 0-13	Se suman a C000 para dar una dirección en el rango C000-FFFF.
Bits 14-15	Un número en el rango 0-3. Este se añade al número de ROM primaria actual para determinar el número de la ROM a la que se accede.

La ROM superior se activa, la ROM inferior se inhibe, se selecciona la ROM superior requerida y se realiza un salto a la dirección especificada. Esta función simplifica el acceso por conmutación entre un grupo superior a cuatro ROM laterales (ROM a la que se accede no directamente, sino con un pequeño rodeo), con números consecutivos, permitiendo con ello la expansión de programas hasta los 64K de memoria.

Debe observarse que si bien INF SALTO no deja dirección de retorno, y es un salto real, no una llamada, por otra parte LLAMADA LATERAL guarda una dirección de retorno que apunta a la posición siguiente a los bytes que produjeron esta llamada, y es una llamada CALL, no un salto. (Las direcciones de retorno dejadas por las construcciones RST se utilizan para localizar los bytes característicos.)

La entrada localizada en 0013 accede mediante un salto a BA10 de las rutinas de la RAM superior. Esta es SALTO HL LATERAL, que se parece a LLAMADA LATERAL, excepto que la palabra de 16 bits que determina el salto está contenida en el registro HL.

La entrada localizada en 0016 accede mediante un salto a la dirección definida por el contenido del registro DE.

Estas dos instrucciones no son accesibles por medio de las instrucciones RST.

RST18 (&DF): entrada localizada en 0018, desde donde se efectúa un salto a B9BF contenida entre las rutinas de la RAM superior. Los dos bytes que acompañan al código &DF se leen como una palabra de 16 bits, que es una dirección que apunta hacia tres bytes característicos. Los primeros dos bytes característicos dan la dirección de una entrada, y el tercer byte es interpretado del siguiente modo:

&00 a &FB	Selecciona la ROM de este número: ROM superior activada, ROM inferior inhibida.
&FC	No se cambia de ROM. ROM superior e inferior activadas.

&FD	No se cambia de ROM. ROM superior activada, la inferior inhibida.
&FE	No se cambia de ROM. Desactivada la superior, activada la inferior.
&FF	No se cambia de ROM. La superior y la inferior inhibidas.

Esta es la denominada LLAMADA, una función versátil que puede acceder a casi todo.

La entrada localizada en 001B accede mediante un salto a B9B1 de las rutinas de la RAM superior. Esta es SALTO HL, que se parece a LLAMADA, excepto en el hecho de que la dirección característica está en el registro HL, mientras que el tercer byte está en el registro C.

La entrada localizada en 001E accede mediante un salto a la dirección definida en el registro HL.

Estas dos entradas no son accesibles por medio de las instrucciones RST.

RST20 (&E7): entrada localizada en 0020, desde donde se efectúa un salto a BACB, rutina contenida entre las de la RAM superior. Esta es LAM RAM, la cual ejecuta una instrucción LD A,(HL) con las ROM inhibidas. Por esta razón puede emplearse para leer la RAM en cualquier momento. El estado en el que la ROM se encontrara anteriormente es restaurado después de la lectura.

La entrada localizada en 0023 accede mediante salto a B9B9 de las rutinas de la RAM superior. Esta es la denominada LLAMADA HL, muy parecida a LLAMADA, pero con la diferencia de que la dirección característica está contenida en el registro HL.

RST28 (&EF): entrada localizada en 0028, desde donde se efectúa un salto a BA2E en la RAM superior. Esta es SALTO ROM INFERIOR. Es semejante a la usual instrucción C3XXXX, diferenciándose en que la ROM inferior se activa antes del salto y se desactiva justo después del retorno.

LLAMADA LATERAL, SALTOHL LATERAL, LLAMADA y LLAMADA HL entran en la rutina llamada, con IY apuntando al área de datos reservada en la RAM por la ROM seleccionada.

RST30 (&F7): es la entrada para RST usuario. Si se utiliza con la ROM inferior activada, el contenido actual de C′, que guarda los bits de selección de la ROM actual, es transferido a (002B) en la RAM, la ROM inferior queda entonces inhibida y la acción retorna a 0030, pero esta vez en RAM. Si la ROM inferior ya estaba previamente inhibida, todo este proceso es innecesario.

El área 0030-0037 de RAM se puede rellenar a voluntad para acceder a rutinas especiales que cumplan los requisitos del usuario. Como en la inicialización 0030 en RAM contiene &C7, si accedemos a esta entrada sin modificarla, provocaremos una reinicialización completa del aparato.

RST38 (&FF): es el equivalente a la respuesta de interrupciones y no está disponible al usuario.

La entrada localizada en 003B es parte del proceso de manipulación de interrupciones. Si una interrupción dura demasiado para tener su origen en el sistema interno, se efectuará una llamada a 003B. Esta contiene normalmente &C9 (instrucción de retorno), pero la RAM desde este punto puede ser modificada para acceder a una interrupción de usuario.

Algunas funciones que han sido descritas raramente serán utilizadas por un típico usuario; aun así, constituyen una parte vital de los sistemas CPC, los cuales no podrían trabajar sin ellas. Su acción se deberá estudiar con cuidado.

Rutinas RAM del bloque de saltos

El bloque de saltos situado al comienzo de la RAM superior ofrece acceso a once funciones:

ACTIVA ROM SUPERIOR *(U ROM ENABLE):* B900, BA5E

La ROM superior seleccionada se activa. Al regreso de la rutina, el registro A contiene el estado previo de ROM.

INHIBE ROM SUPERIOR *(U ROM DISABLE):* B903, BA68

La ROM superior seleccionada se inhibe. Al regreso de la rutina, el registro A contiene el estado previo de ROM.

ACTIVA ROM INFERIOR *(L ROM ENABLE):* B906, BA4A

La ROM inferior se activa. La rutina regresa con A conteniendo el estado previo de ROM.

INHIBE ROM INFERIOR *(L ROM DISABLE):* B909, BA54

La ROM inferior se inhibe. La rutina regresa con A conteniendo el estado previo de ROM.

Estas cuatro rutinas son casi idénticas, tomando la forma general siguiente:

Prohíbe interrupciones
Selecciona los registros alternativos
A=C′
Modifica C′
OUT (C′),C′
Selecciona los registros normales
Habilita interrupciones
Retorno

La modificación de C′ afecta al bit 2 para la ROM baja y al bit 3 para la ROM alta. El bit relevante estará puesto a cero para activar y a uno para inhibir. La salida efectuada se dirige al controlador de video. Observa que se asume que B′ contiene &7F, byte superior de la dirección requerida de E/S. Los contenidos de B′ sólo se pueden cambiar partiendo de una base temporal, es decir, mientras las interrupciones no estén permitidas y no se realicen llamadas del sistema operativo.

RESTAURA ROM *(ROM RESTORE):* B90C, BA72

Al comienzo, A debe contener los bits del estado requerido de la ROM, tal como se definió antes, proporcionados por las cuatro rutinas anteriores. Este estado se coloca entonces.

La rutina es similar a las cuatro primeras, excepto que los bits 2 y 3 de A son transferidos a C′, quedando los restantes bits de este último registro inalterados.

SELECCIONA ROM *(ROM SELECT):* B90F, BA7E

Al comenzar, C contiene el número de la ROM requerida. Esta ROM es seleccionada y la ROM superior queda activada. Al regreso de la rutina llamada, C contiene el número de la ROM seleccionada antes de la actual y B contiene los bits de estado.

La rutina efectúa una llamada a ACTIVA ROM SUPERIOR; después salta a BA92, desde donde una operación de salida de C hacia DFXX selecciona la ROM requerida. El número de la ROM se copia en (B1A8), que mantendrá los datos de la ROM superior que se esté utilizando actualmente.

ROM ACTUAL *(CURR SELECTION):* B912, BAA2

El registro A se carga desde (B1A8) con el número de la ROM actual.

TIPO ROM *(PROBE ROM):* B915, BAA2

Al comienzo, C contiene una dirección de la ROM seleccionada. A la salida, A contiene el tipo de ROM, H contiene el número de versión de la ROM y L contiene el número de marca de la ROM. El byte que describe el tipo se interpreta así:

0	ROM primaria
1	ROM secundaria
2	Extensión de la ROM primaria
&80	Es la propia del ordenador.

La rutina realiza una llamada a SELECCIONA ROM, para recoger los datos de la ROM seleccionados, entonces A=(C000), HL=(C001/2). ROM ANTERIOR viene a continuación para restaurar la ROM presente anteriormente.

ROM ANTERIOR *(ROM DESELECT):* B918, BA8C

Al comienzo, C contiene el número de la ROM requerida y B contiene el estado de dicha ROM. Normalmente, éstos dos datos han sido obtenidos por SELECCIONA ROM. La ROM especificada y su estado se seleccionan mediante RESTAURA ROM y la rutina contenida en BA92, que era utilizada por SELECCIONA ROM.

LDIR: B91B, BAA6

LDDR: B91E, BAAC

Estas rutinas permiten transferir datos desde la RAM a la propia RAM, con la ROM temporalmente inhibida. Al comenzar, BC, HL y DE deben estar inicializados como si fuera para una instrucción LDIR o una LDDR.

Las rutinas son bastante enrevesadas y únicamente se pueden seguir observando cómo cambian los contenidos de la pila. Aquí viene presentada la rutina LDIR. La rutina LDDR es casi idéntica:

```
BAA6    CALL BAB2         Pila: X
BAA9    LDIR              Pila: X, BC', BABF
BAAB    RET               Hacia BABF

BAB2    DI                Pila: X, BAA9
        EXX
        POP HL'           Pila: X HL'=BAA9
        PUSH BC'          Pila: X,BC'
        C'=C' OR &0C
        OUT (C'),C'       Inhibe ROMs
        CALL BAC7         Pila: X, BC'

BABF    DI                Pila: X, BC'
        EXX
        POP BC'           Pila: X
        OUT (C'),C'       Restaura ROM
        EXX
        EI
        RET               Hacia X (Dado por la rutina que hizo
                          la llamada)

BAC7    PUSH HL'          Pila: X, BC', BABF, BAA9
        EXX
        EI
        RET               Hacia BAA9
```

Area de extensiones RST

Las rutinas utilizadas para implementar las funciones del área RST son complejas, además de dar muchos rodeos, pero es aconsejable revisarlas con cierto detalle para que su acción quede claramente entendida. Estas rutinas se examinarán en el mismo orden en el que aparecen en la RAM superior.

INF SALTO HL *(LOW PCHL):* 000B, B97C

```
B97C  DI
      PUSH HL
      EXX
```

```
        POP DE'
        JP B988                 Véase más adelante
```

La palabra (16 bits) característica es transferida de HL a DE'.

INF SALTO *(LOW JUMP):* RST08, B982

```
B982    DI
        EXX
        POP HL'                 Dirección de retorno
        DE'=(HL')               Palabra característica en DE'

B988    EX AF/AF'
        A'=D'                   Byte superior de la palabra
        D'=D' AND &3F           DE' contiene únicamente la dirección
        RLCA
        RLCA                    Bits 6, 7 a 0, 1

B990    RLCA
        RLCA                    Bits 0, 1 a 2, 3
        XOR C'
        AND &0C                 Bits 2, 3 de A' aislados
        PUSH BC'                Guarda el estado anterior de la ROM
        CALL B9A8               Véase debajo
        DI
        EXX
        EX AF/AF'
        A'=C'
        POP BC'
        AND 3
        C'=C' AND &FC
        A'=A' OR C'
        JP B9A9

B9A8    PUSH DE'                Enlaza con la rutina llamada
B9A9    OUT (C'),C'             Activa según estado previo
        CLEAR CARRY'            Acarreo a cero
        EX AF/AF'
        EXX
        EI
        RET
```

La manipulación de las direcciones de retorno son de alguna manera semejantes a aquellas observadas anteriormente en LDIR/LDDR. El pun-

to crucial es si el último bloque debe comenzar en B9A8 o en B9A9. Para B9A8, la dirección requerida de entrada se coloca en la pila, por lo que el bloque regresará a la rutina llamada. Si se comienza en B9A9, se dejará la dirección global de retorno en la pila. Por eso, el mismo bloque realiza dos cosas totalmente diferentes.

La rutina es complicada por la necesidad de preservar los restantes bits de C′ mientras se manipulan los bits 2 y 3 para seleccionar el estado de la ROM. Por otra parte, los bits 0 y 1 pueden ser modificados durante la ejecución de la rutina llamada, pero se deben incorporar de nuevo para formar con los otros bits el byte de estado previo.

SALTO HL *(FAR PCHL):* 001B, B9B1

```
B9B1  DI
      EX AF/AF'
      A'=C            Número de ROM
      PUSH HL         Dirección rutina
      EXX
      POP DE'         Dirección rutina en DE'
      JP B9CE         Véase debajo
```

LLAMADA HL *(FAR ICALL):* 0023, B9B9

```
B9B9  DI
      PUSH HL         Puntero hacia palabra característica
      EXX
      POP HL'         Puntero hacia HL'
      JP B9C8         Véase debajo
```

LLAMADA *(FAR CALL):* RST18, 0018, B9BF

```
B9BF  DI
      EXX
      POP HL'         Retorno del enlace
      DE'=(HL')       Puntero
      HL=HL+2
      PUSH HL HL'     Modifica retorno del enlace
      EX DE'/HL'      HL' contiene el puntero
B9C8  DE'=(HL')       Dirección
      HL'=HL'+2
      EX AF/AF'
      A'=HL'          Número de ROM
```

```
B9CE    IF A'> &FB THEN B990
B9D2    B'=&DF
        OUT (C'),A'                 Selecciona ROM
        B'=(B1A8)                   Número de ROM anterior
        (B1A8)=A'                   Nuevo número de ROM
        PUSH BC'
        PUSH IY
        A'=A'-1
        IF A')6 THEN B9F2           No es ROM secundaria
        HL'=B1AC+2*A'
        IY=(HL')                    Dirección de espacio de trabajo
B9F2    B'=&7F                      Restaura valor normal
        A'=C' AND &F3
        CALL B9A8                   Accede a rutina llamada
        POP IY                      Restaura los contenidos an-
                                    teriores
        DI
        EXX
        EX AF/AF'
        E'=C'                       Guarda valor actual
        POP BC'                     Restaura antiguo valor
        A'=B'                       ROM anterior
        B'=&DF
        OUT (C'),A'                 Restaura ROM anterior
        (B1A8)=A'                   Observa ROM actual
        B'=&7F                      Restaura valor normal
        A'=E'                       ROM que fue llamada
        JP B99F                     Véase arriba
```

SALTO HL LATERAL *(SIDE PCHL):* 0013, BA10

```
BA10    DI
        PUSH HL                     Palabra característica
        EXX
        POP DE'                     Palabra en DE'
        JP BA1E                     Véase debajo
```

LLAMADA LATERAL *(SIDE CALL):* RST10, 0010, BA16

```
BA16    DI
        EXX
        POP HL'                     Retorno del enlace
```

```
        DE'=(HL')            Palabra característica
        HL'=HL'+2
        PUSH HL'             Modifica retorno del enlace
BA1E    EX AF/AF'
        A'=D'                Byte superior de la palabra
        D'=D' OR &C0         Prepara dirección entre C000-FFFF
        A'=A' AND &C0        Aísla los bits de selección de ROM
        RLCA
        RLCA                 Bits 6, 7 a 0, 1
        A'=A'+(B1AB)         Añade número de ROM primaria
        JP B9D2              Véase arriba
```

SALTO ROM INFERIOR *(FIRM JUMP):* RST28, 0028, BA2E

```
BA2E    DI
        EXX
        POP HL'              Apunta hacia la palabra característica
        DE'=(HL')            Palabra característica
        C'=C' AND &FB        Bit 2 a cero
        OUT (C'),C'          Activa ROM inferior
        (BA3F/40)=DE'        Modifica instrucción
        EXX
        EI
BA3E    CALL XXXX            Dirección definida antes
        DI
        EXX
        C'=C' OR 4           Bit 2 a uno
        OUT (C'), C'         Inhibe ROM inferior
        EXX
        EI
        RET
```

Esta rutina implica la creación de una instrucción en tiempo real, lo cual no se aprecia por todos los programadores, pero funciona. Sin embargo, se pueden producir resultados confusos al desensamblar.

LAM RAM: RST20, 0020, BACB

```
BACB    DI
        EXX
        E'=C'                Estado activo
```

```
        E'=E' OR &0C
        OUT (C'), E'            Inhibe ambas ROM
        EXX
        A=(HL)                  Lee de la RAM
        EXX
        OUT (C'), C'            Restaura el estado activo
        EXX
        EI
        RET
```

La rutina final no tiene nombre, pero es una variación de LAM RAM:

```
BADC    EXX
        A=C' OR &0C
        OUT (C'), A             Inhibe ROM
        A=(IX)'                 Lee RAM
        OUT (C'), C'            Activa ROM
        RET
```

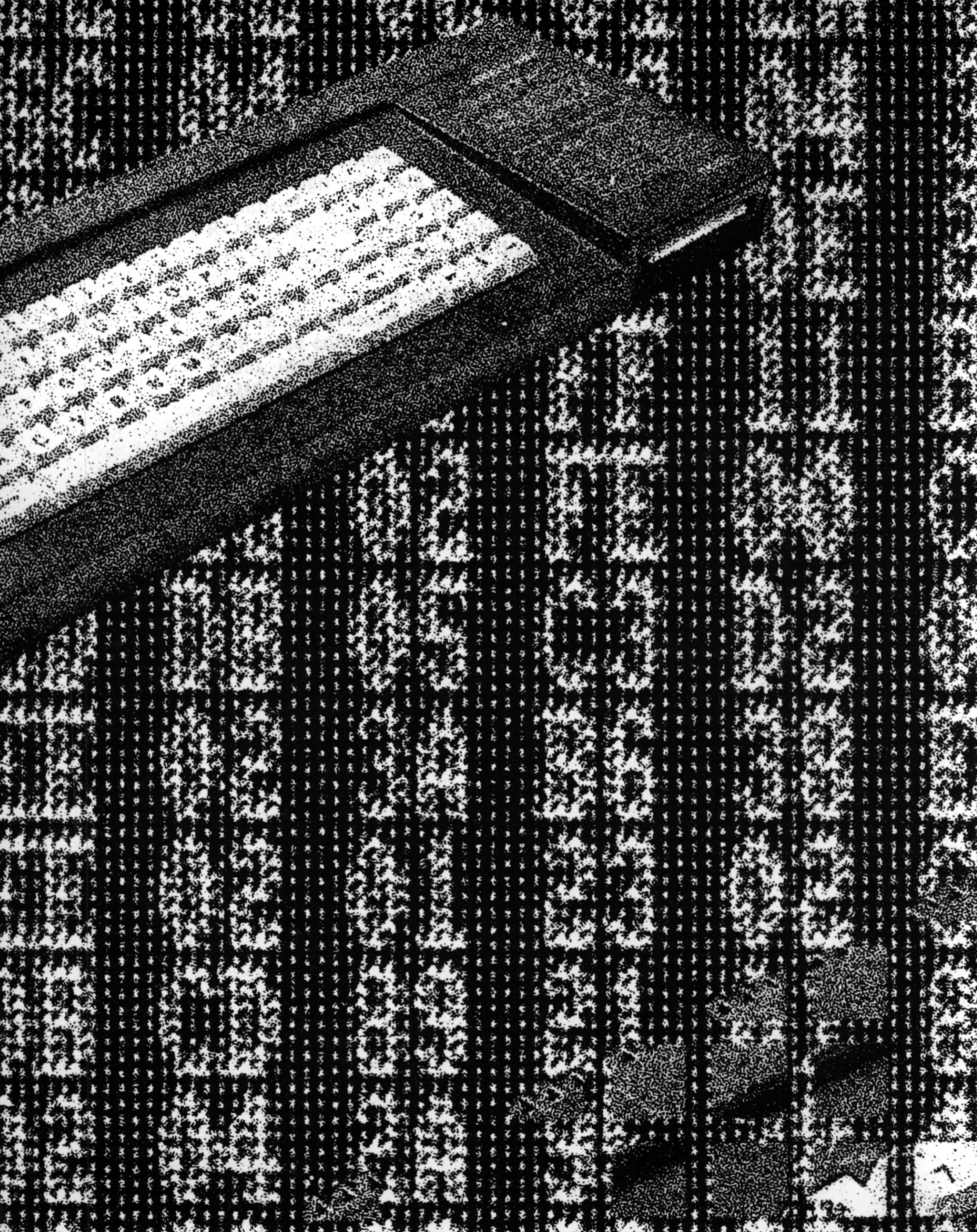
De esta manera no se altera E' y se utiliza IX como puntero en vez de HL.

Comentarios

Un programador experimentado, echando un vistazo al contenido de este capítulo, sacudiría amenazadoramente la cabeza. Su opinión inicial sería que sobra mucha redundancia (¡menudo jaleo!), pero después de un estudio completo llegaría a la conclusión de que cada rutina es necesaria para implementar el sistema de almacenaje con todas sus posibilidades. Un usuario que no observe este detalle encontrará que las rutinas hacen que todo sea sencillo, y éste es el punto clave.

Por ejemplo, mientras el intérprete de BASIC esté trabajando, la ROM superior debe estar activada, pero el programa BASIC está almacenado desde 0170 en adelante, bajo la ROM inferior. LAM RAM permite acceso directo a esta área de la RAM, con un mínimo de molestia y complicación. Si se necesita una función de la ROM inferior, se puede llamar a través del bloque de saltos por RST8 o RST28.

Las rutinas que han sido descritas forman la base del sistema operativo CPC464 y CPC6128, una creación alrededor de la cual se construye el resto del sistema. Ahora podemos continuar examinando las partes más altas y directamente más interesantes del edificio.



3
El control de la máquina

Definiéndolo por encima, este grupo de control es el responsable de dirigir los periféricos *hardware*, aunque será conveniente incluir el proceso de inicialización principal bajo este título, ya que está referido ampliamente a los de inicialización de periféricos.

Muchas de las rutinas del grupo de control dependen de la acción de otras rutinas de actualización de datos. Para entender por completo estos datos, necesitas leer *Programación avanzada del Amstrad**, que ofrece todos los detalles de los códigos de los periféricos. Unicamente se definirán aquí los códigos más esenciales.

Reinicialización principal

Al conectar el aparato, o como respuesta al código de instrucción &C7, se direcciona la entrada localizada en 0000. En el momento de la conexión, la ROM inferior está activada, pero, si se trata de una llamada RST0, el estado de la ROM carecerá de importancia, ya que en ese momento esta zona de la RAM estará cargada con los mismos datos de la ROM. Ya sea desde la ROM o desde la RAM, la primera acción de la rutina de reinicialización es introducir el byte &89 en la puerta de ordenación de video, junto a una dirección 7FXX de E/S. Con esto se activa la

* Thomasson, Don: *Programación avanzada del Amstrad (Descripción de la ROM. Rutinas y parámetros)*, Anaya Multimedia, Madrid, 1985.

ROM inferior, aunque ya estuviese en ese estado, se inhibe la ROM superior, y también se fija el modo 1. Ya no se efectuarán más acciones desde el área RST, y la rutina salta a 0580 para continuar la acción de inicialización.

Se prohíben las interrupciones y el byte &82 se envía junto a 7FXX. Esto fijará la actuación del PPI (interfaz periférico paralelo), imponiendo como salidas los puertos A y C, y como entrada el puerto B. Enviando F4XX y F6XX acompañadas del byte &00, se pondrán a cero los puertos A y C, mientras que enviando EFXX junto al byte &7F se consigue inicializar la puerta de impresora. El bit 7 queda a nivel bajo y el resto de los bits a nivel alto.

El controlador CRT se actualiza entonces. Existen dos valores alternativos para esta acción. Uno para una frecuencia de red de 50 Hz y el otro para 60 Hz. El valor a utilizar está determinado por el bit 4 de la puerta B del controlador. Si este bit está puesto a 1, se empleará el valor de 50 Hz, mientras que el valor de 60 Hz se usará si el bit está puesto a cero. El valor de este bit se determina por la presencia de la conexión 4 con el circuito principal de la impresora. Las tablas de este circuito se leen en sentido inverso, lo cual puede ser un poco confuso al principio. Los valores de salida oscilarán entre BCXX, que selecciona el registro a actualizar, y BDXX, que realiza esa actualización.

A continuación se llama a MC EJECUTA PROGRAMA situada en 060E con DE=065C y HL=0000. El contenido de DE apunta a la rutina de pantalla que escribe el título de presentación, y que es llamada en el momento apropiado. El valor cero en HL significa que la ROM 0 comienza en C006. Normalmente aquí está el intérprete de BASIC, a no ser que una ROM externa responda al valor 0. C006 es el valor estándar utilizado como punto de comienzo de la ROM superior.

Antes de discutir acerca de MC EJECUTA PROGRAMA, será conveniente estudiar un programa que llama a esta rutina durante su ejecución, habiendo cargado primero los datos necesarios:

MC CARGA Y EJECUTA *(MC BOOT PROGRAM):* BD13, 05DC

Al acceder a esta función, HL debe contener la dirección de una rutina de carga, la cual debe estar diseñada de forma que regrese con el acarreo a uno y la dirección de comienzo del programa en HL si la carga se efectuó correctamente, o con el acarreo a cero si la carga falló.

La pila se inicializa con SP=C0000, siendo ésta la posición habitual de la pila. Se llama a REINICIALIZA SONIDO para silenciar al generador de sonido. Se prohíben las interrupciones y se envía &FF al puerto F8FF, requiriendo con ello que todos los dispositivos periféricos externos sean reinicializados.

A continuación se llama a NC REINICIALIZA NUCLEO para poner a cero el área B100-B1BF, salvando antes de esto el contenido previo de (B1A9/B). (B1A9/A) contiene la dirección de entrada de la última ROM primaria utilizada; este contenido es copiado en DE, mientras que (B1AB) contiene el número de la última ROM primaria utilizada, siendo dicho contenido copiado en B. (Observa que el número de la ROM actualmente en uso está contenido en (B1A8) y que no es preservado en esta ocasión.) Si (B1AB) contiene &FF, la rutina regresará con C, D y E puestos a cero.

De regreso ya en rutina principal de CARGA y EJECUTA, HL es restaurado con el valor que tenía al principio y DE, BC y HL son entonces guardados en la pila. Se llama a TCL RE-INICIALIZACION para inicializar el teclado; a continuación se llama a TXT REINICIALIZA para inicializar la pantalla de texto, asistida por una llamada a PNT REINICIALIZA. Se llama después a ACTIVA ROM SUPERIOR para hacer que la ROM superior entre en acción.

Se recupera HL de la pila, y se accede al programa cargado, que está definido por HL, mediante una pequeña y aislada subrutina que consiste solamente en la instrucción JP (HL). BC y DE se recuperan entonces de la pila.

Si la rutina de carga regresa con el acarreo a uno, entonces MC EJECUTA PROGRAMA se inicia en 060B. En caso contrario, se intercambian DE y HL, poniendo la dirección obtenida por NC REINICIALIZA NUCLEO en el registro HL, C=B, y MC EJECUTA PROGRAMA se inicia en 060E con DE=06E8, que es la dirección de comienzo de una rutina que informa "LOAD FAILED" (fallo en la carga del programa). La ROM anteriormente seleccionada se direcciona entonces y a ella se le deja el control.

MC EJECUTA PROGRAMA *(MC START PROGRAM):* BD16, 060B

Si se utiliza la dirección normal de entrada a esta función, esto es, 060B, entonces DE es cargado con 0726 (apuntando a una instrucción de retorno), pero también es posible comenzar en la dirección 060E, con DE apuntando a una subrutina que se ejecutará en la última parte de la rutina EJECUTA PROGRAMA. En ambos casos, HL debe contener la dirección de comienzo del programa y C debe contener el número de la ROM a emplear, aunque este contenido pueda ser irrelevante en caso de que HL apunte a un área de la RAM.

Se prohíben las interrupciones y se selecciona el modo 1 de interrupciones. Los registros alternativos BC, DE y HL se introducen en la acción.

Una salida de 0 en la dirección DFXX selecciona la ROM 0 superior, y una salida del byte &FF, junto a la dirección F8XX de E/S, reinicializará los periféricos externos. El espacio de trabajo comprendido entre B100 y

B8FF se pone a cero, y la puerta de ordenación de video recibe una salida de &89 con la dirección 7FXX. (Modo 1 activa ROM inferior e inhibe la superior.) Los registros normales BC, DE y HL son de nuevo seleccionados. Con esto se fijan las condiciones iniciales requeridas por el sistema de interrupciones.

La pila se actualiza de nuevo con C000, su estado base, y los registros HL, BC y DE se guardan en la pila. Una serie de llamadas posteriores realizan la inicialización principal.

En 0044, se copian las rutinas de RAM desde la ROM, con una llamada a NC REINICIALIZA NUCLEO. A continuación:

RESTAURA BLOQUE SALTOS que reinicializa las entradas de este bloque.
NC INICIALIZACION que reinicializa el teclado.
REINICIALIZA SONIDO para el sistema de sonido.
TXT INICIALIZA inicializa la unidad de video (VDU) para texto.
GRA INICIALIZA inicializa la unidad de video (VDU) para gráficos.
CAS INICIALIZA para el sistema del cassette.
MC INICIALIZA IMPRESORA que estandariza el sistema de impresora.
PNT INICIALIZA que se encarga del control de pantalla.

Los detalles de estas rutinas serán examinados en la sección correspondiente, pero se puede decir que todo —o casi todo— es llevado a su estado básico. Esto puede ser molesto para alguien que quiera una actualización de condiciones no estándar, pero tiene la gran ventaja de que cada programa comienza sobre una misma base.

Después de todas estas llamadas, se activan las interrupciones, y se llama la rutina definida al comienzo en DE. Esta puede ser la que presenta en la pantalla el título inicial, o un mensaje *load failed* como el definido anteriormente. El programa principal recupera entonces BC y HL de la pila y efectúa un salto a 0077, que es la entrada actual de la rutina. Esta rutina clave no es accesible por medio del bloque de saltos, lo que podría ser muy útil, debido a que, según la concepción del sistema, se requiere una reinicialización completa antes de comenzar ningún programa.

Si HL contiene 0000, ejecuta la entrada prescrita para omisiones y localizada en la dirección C006 de la ROM 0; en otro caso, se ejecutará la dirección de la entrada definida por HL de la ROM indicada por el contenido de A.

0077	Si HL=0000, HL=C006, A=0	Valor para omisiones
	(B1A8)=A	Número de ROM
	(B1AB)=A	Byte característico
	(B1A9/A)=HL	Dirección característica
	HL=ABFF	Valor inicial de HIMEM
	DE=0040	Valor inicial de la dirección

	más baja donde se pueden guardar programas.
BC=B0FF	Tope de la memoria utilizable.

Una LLAMADA con DF A9 B1 accede a la rutina.
Al regreso, se ejecuta una reinicialización completa desde 0000.

Con esto se completa la rutina MC EJECUTA PROGRAMA, aparte de las rutinas llamadas al final:

065C: desde donde se llama a 0712, que lee el puerto B, de la que se toman los bits 1-3, que son determinados por las conexiones. De acuerdo con el tipo de conexión realizada, en la pantalla se anuncia que el nombre del aparato es uno de los siguientes:

Arnold	Amstrad	Orion
Schneider	Awa	Solovox
Saisho	Triumph	Isp

La salida actual a la pantalla es manipulada por 06EB, que también es llamada con HL=066D para sacar el resto del título, y, finalmente, con HL=0693 para completar la presentación.

El conjunto 06E8 con HL=06F4 apuntan al mensaje "***PROGRAM LOAD FAILED***".

El contenido de la entrada en 06EB es el siguiente:

```
06EB  A=(HL)
      HL=HL+1
      Si A=0 entonces regresa
      CALL TXT SALIDA
      JP 06EB
```

Rutinas de impresora

El acceso a la puerta de la impresora se obtiene mediante un pequeño grupo de rutinas muy relacionadas.

MC INICIA IMPRESORA *(MC RESET PRINTER):* BD28, 07E6

Es la indirección para la entrada situada en BDF1, MC ESPERA IMPRESORA y se reinicializa para acceder a 07F8. Esto cancela cualquier cambio que haya sido efectuado para obtener un controlador alternativo de la impresora.

MC IMPRIME CARACTER *(MC PRINT CHAR):* BD2B, 07F2

BC se guarda en la pila a la vez que se efectúa una llamada a MC ESPERA IMPRESORA localizada en BDF1.

MC ESPERA IMPRESORA *(MC WAIT PRINTER):* BDF1, 07F8

BC se carga con (0032), actuando como contador de retardo. Se efectúa una llamada a MC IMPRESORA OCUPADA, y si al regreso de ésta el acarreo está a cero, entonces la rutina salta a MC ENVIA CARACTER. Esto significa que la impresora no está ocupada.

Si el retorno se efectúa con el acarreo a uno, la rutina volverá a repetir la llamada estableciéndose un bucle. En total, la llamada se ejecutará 12.800 veces antes de abandonar y regresar con el acarreo a cero para indicar el fallo. Esto te da un amplio margen de tiempo para poner la impresora en la línea si habías olvidado hacerlo.

MC ENVIA CARACTER *(MC SEND PRINTER):* BD31, 0807

BC, conteniendo el retardo, se guarda en la pila y A AND &7F se envía a la puerta de la impresora en la dirección EFXX. Con ello se pone al bit STROBE a nivel bajo. Después se envía A OR &80 a la misma dirección, lo que hará ponerse a nivel alto al bit STROBE. Finalmente, A AND &7F es enviado de nuevo para poner a nivel bajo la línea STROBE. Durante el último par de envíos, se inhiben las interrupciones para evitar el riesgo de una duración excesiva de la señal STROBE. BC es recuperado, el acarreo es puesto a 1, y la rutina retorna.

MC IMPRESORA OCUPADA *(MC BUSY PRINTER):* BD2E, 081B

BC, conteniendo la cuenta del retardo, es guardado en la pila y A es transferido al registro C. Entonces se carga A con una entrada del puerto B en F5XX, y el bit 6, que es la línea BUSY (ocupada) de la impresora, es transferido al bit de acarreo. A es restaurado desde C, BC es recuperado de la pila y se efectúa el retorno de la rutina.

Este es un buen momento para recordarte cuál es la diferencia clave entre las entradas del bloque de saltos principal y las indirecciones. Aparte de la preservación de BC en el primer caso, BD2B y BDF1 parecen tener el mismo efecto, pero BD2B activa la ROM inferior, y BDF1 no lo hace. La utilidad de hacer a BDF1 una indirección, es simplificar las llamadas a controladores alternativos, pero hacer esta llamada con la ROM inferior inhibida puede provocar un caos. Sin embargo, si la indirección ha sido alterada para llamar a una rutina en la RAM, este caos no se producirá.

Otras rutinas MC

MC BORRA TINTAS *(MC CLEAR INKS):* BD22, 0786

BC y DE se guardan en la pila y BC=7F10 forma una dirección de la matriz de puertas de video. Se llama a 07AB (véase debajo).

En 0790, se vuelve a llamar a 07AB otra vez. DE es decrementado y, si al regreso de 07AB el bit Z está puesto a cero, se inicia un bucle volviendo a 0790. En caso contrario, BC y DE son recuperados y la rutina retorna.

Como en 07AB se incrementa DE, el contenido de este registro permanece igual durante la ejecución del bucle.

MC ASIGNA TINTAS *(MC SET INKS):* BD25, 0799

Esta rutina es idéntica a MC BORRA TINTAS, excepto en que se omite el decremento de DE. Con ello se permite el incremento de DE en 07AB.

En 07AB se fija el puntero de la paleta de la matriz de puertas de video mediante una instrucción OUT (C),C. Después se envía a este mismo puerto la combinación A=(DE) AND &1F OR &40 para poner dicha paleta de tintas como entrada. DE y C son incrementos, y si C=&10 el bit de cero (Z) se coloca a uno. La rutina regresa entonces.

Las rutinas anteriores precisan comenzar con DE apuntando a una entrada en la tabla de colores, la cual será discutida posteriormente. MC BORRA TINTAS coloca todas las entradas de la paleta a un mismo color; MC ASIGNA TINTAS coloca las entradas de la paleta según la tabla de colores.

MC ESPERA BARRIDO *(MC WAIT FLYBACK):* BD19, 07BA

AF y BC se guardan en la pila, y se prepara el acceso al puerto B haciendo B=&F5. Se toma un byte de entrada del puerto, y el bit de acarreo toma el valor del bit 0 de esta entrada. Esta operación se repite hasta que el acarreo resulte ser uno, lo que indicará que ha finalizado el barrido de la pantalla. BC y AF son entonces recuperados y la rutina regresa.

Esta rutina permite retrasar cualquier acción sobre la pantalla hasta que finalice la reproducción de la nueva (o misma) imagen.

MC OFFSET PANTALLA *(MC SCREN OFFSET):* BD1F, 07C6

Al comienzo, A debe contener el byte superior de la base requerida de pantalla, que está contenido en (B1CB) y es colocado allí por PNT IMPONE BASE RAM. El par HL debe contener el desplazamiento de pantalla respecto a la base, como el contenido en (B1C9/A).

- BC se guarda en la pila.
- C=A/4 AND &30.
- A=H/2 AND 3 OR C.
- Salida de &0C a BCXX para seleccionar el registro 12 del controlador CRT.
- Salida de A a BDXX para actualizar este registro.
- Salida de &0D a BCXX para seleccionar el registro 13 del controlador CRT.
- HL=HL/2.
- Salida de L a BDXX para actualizar este registro.
- BC es recuperado.
- Retorno.

Este es un ejemplo de una rutina MC que ayuda a implementar una rutina en otra parte, ajustando el *hardware* según los ajustes del *software*. Los valores fijados pueden parecer extrañamente distorsionados, hasta que la sección de la pantalla haya sido leída.

MC MODO PANTALLA *(MC SET MODE):* BD1C, 0776

Al comienzo, A debe contener el número requerido para el modo. Si el contenido de A excede de 2, la rutina finalizará.

Los bits 0 y 1 de A se copian en los correspondientes bits de C', y entonces C' se envía a 7FXX, la matriz de puertas de video.

Puede producirse una seria confusión si esta función no fue ejecutada después de un cambio de modo *software*. El *hardware* y el *software* deben mantenerse en buenas relaciones para que todo funcione.

MC REGISTRO SONIDO *(MC SOUND REGISTER)*: BD34, 0826

La rutina transfiere datos a los registros del generador de sonido. Los datos se transfieren a través del puerto A del PPI, mientras que la interpretación de los datos se controla por los bits 6 y 7 del puerto C, de esta forma:

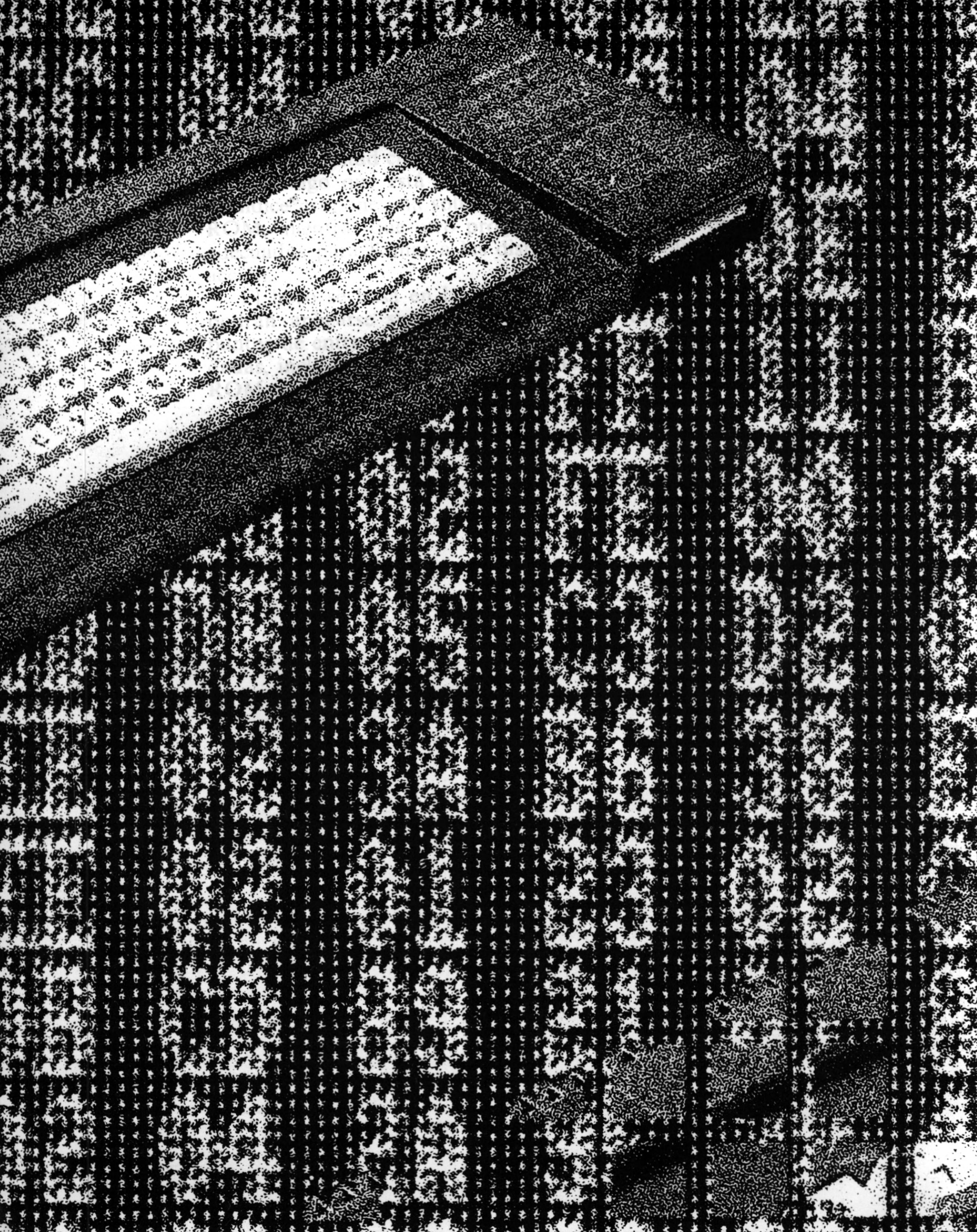
Bit 6	*Bit 7*	
0	0	Inactivo
0	1	Escribe en el registro
1	0	Lee del registro
1	1	Selección de registro

Al comienzo de esta rutina, A debe contener el número de registro y C debe contener el dato. Las interrupciones son inhibidas.

A es enviado a F4XX	El puerto A especifica el registro
A se carga desde F6XX	Entrada puerto C
A=A OR &C0	Bits 6 y 7 a uno
A es enviado a F6XX	Selecciona registro
A=A AND &3F	Bits 6 y 7 a cero
A es enviado a F6XX	Inactivo
C es enviado a F4XX	Puerto A especifica el dato
C=A	
A=A OR &80	Bit 7 a uno
A es enviado a F6XX	Escribe en el registro
C es enviado a F6XX	Inactivo
Interrupciones permitidas	
Retorno	

Esta rutina sólo maneja envíos o salidas al generador de sonido. Las entradas a través del teclado se manipulan en otra parte.

Con esto se completa el estudio de las rutinas de control de la máquina.



4
El núcleo

Las rutinas del núcleo se encargan de las interrupciones y de los sucesos que deben ocurrir cada cierto tiempo. Son bastante complejas y enrevesadas, pero es necesaria su comprensión para toda persona que desee utilizar a fondo el sistema CPC464 y CPC6128. Lo mejor será empezar viendo rápidamente por encima la disposición del sistema.

La matriz de puertas de video genera un pulso de interrupción cada 1/300 segundos, o para ser más precisos, cada 52 barridos horizontales del haz de pantalla. Esto significa un intervalo de 64*52=3328 microsegundos.

El procesador responde al pulso de interrupción con un salto a la dirección 0038, desde donde se efectúa un salto al manipulador primario de interrupciones, situado entre las rutinas de la RAM en B939. El manipulador primario llama a un secundario situado en ROM en la dirección 00B1, y lo primero que éste hará será actualizar el contador de tiempo. Después, y si los hay, se encarga de los sucesos asociados síncronamente con el barrido de pantalla y, finalmente, llamará a la rutina de interrupción del sistema de sonido. Estas son las funciones de las interrupciones de "temporización rápida".

En cinco de cada seis casos, la rutina secundaria finaliza y regresa, completándose así el proceso de manipulación. Pero en uno de cada seis casos la acción continúa para dar servicio a las interrupciones de "temporización lenta", que nominalmente ocurren cada 1/50 segundos. En este caso,

el manipulador principal llama a otro manipulador secundario, localizado en ROM en la dirección 010A.

Debido a que las interrupciones se producen tan frecuentemente, es esencial que sean manejadas tan rápidamente como sea posible, ya que este tiempo se está robando a las rutinas principales que se estén ejecutando en ese momento. Incluso así, no siempre hay tiempo para completar todas las acciones preestablecidas antes de que tenga lugar la siguiente interrupción. Por ello se hace un pequeño almacén provisional para aquellas acciones que hayan tenido que ser abandonadas ante la imposibilidad de su realización.

Existe un almacén provisional para el manejo de interrupciones especiales del usuario, pero eso lo veremos más adelante.

La mayoría de las acciones inducidas por interrupciones son sucesos, cada uno de los cuales está identificado por un bloque de sucesos que define sus características y la dirección de la rutina que los implementa. Los sucesos pueden ligarse a la temporización rápida, a la temporización lenta o a las interrupciones asociadas con el barrido de pantalla. Incluso pueden ser activadas por el programa principal.

El controlador de interrupciones

MC EJECUTA PROGRAMA selecciona el modo 1 de interrupciones, lo que significa que el procesador responde a la interrupción, poniendo en la pila la dirección de la siguiente instrucción a ejecutar y saltando seguidamente a 0038, desde donde se salta a B939 en las rutinas de la RAM.

El primer requisito para un controlador de interrupciones es preservar el contenido de los registros del procesador, de forma que la rutina interrumpida pueda continuar cuando el controlador haya completado su labor. Comúnmente esto se lleva a cabo poniendo en la pila el contenido de cada uno de los registros, pero los CPC464 y CPC6128 utilizan un método más rápido, que consiste en conmutar los registros alternativos, al menos inicialmente. Aquí comienzan una especie de ejercicios gimnásticos.

Primero se selecciona AF′. Si el acarreo alternativo C′ está puesto a 1, la rutina salta a B970. El sistema ya está en el curso de la interrupción y se necesita una acción especial, como se explica más adelante. Si C′ se encontraba a cero, entonces se pone en actividad a BC′, DE′ y HL; para preservar el estado de la ROM actualmente activa se hace A′=C′ y se pone el acarreo C′ a uno.

Las interrupciones se habilitan brevemente mientras se selecciona AF. Este es el único escaso período de tiempo durante la ejecución del controlador en el que se puede introducir una segunda interrupción. Esto ocurre 44 ciclos de reloj después de que la interrupción tenga lugar, es decir, 13 microsegundos. Si la interrupción original está todavía activa, querrá decir que no procede de la matriz de puertas de video; por tanto, se trata de

una interrupción del usuario. Como el acarreo C′ está a uno, tendrá lugar el salto a B970 mencionado anteriormente. Después veremos las consecuencias de esto.

AF se guarda ahora en la pila para conservar su contenido y el bit 2 del registro C′ se pone a cero, de forma que la siguiente instrucción OUT (C′), C′ activará la ROM inferior. Con esta acción ya es posible llamar a 00B1 en la ROM. Allí se encuentra el primer manipulador secundario.

En 00B1, se decrementa el contador de tiempo situado en (B187/B). Después se toma un byte de entrada desde F5XX (puerto B). Si el bit 0 está a uno, es un tiempo de barrido de pantalla y, si hay algún suceso asociado síncronamente al barrido de pantalla, éste será atendido mediante una llamada a 0153 con HL=(B1BC) (véase "Sucesos").

Si hay algún suceso en la lista de la temporización rápida, será entonces atendido llamando a 0153 con HL=(B18E). Esto completa las acciones que ocurren 300 veces por segundo.

Ahora se decrementa el contador situado en (B192), y si el resultado es distinto de cero, la rutina retorna. En caso contrario, (B182) se reinicializa a 6 y son ejecutadas las acciones de temporización lenta.

Primero se llama a la rutina de exploración de teclado (véase "Monitor de teclado"). Después se comprueba la lista de sucesos de temporización lenta. Si no está vacía, el bit 6 de (B104), que es un byte que actúa como *flag*, se pone a uno. La rutina regresa entonces.

De vuelta en el controlador principal, se llevan a cabo unos cuantos juegos malabares. El acarreo se pone a cero y se convierte en acarreo alternativo mediante EX AF,AF′. El registro A contiene ahora el estado de la ROM anterior, que se copió al principio de la rutina desde C′ y se introdujo en A′. C′=A′ y B=&7F, que es su valor usual.

Si (B104)=0 o si (B104) es negativo, la rutina saltará a B96A. En otro caso, A=C′ AND &0C y se guarda AF en la pila. Se pone a cero el bit 2 del registro C′. Se seleccionan los registros normales BC, DE y HL y se llama a 010A para ejecutar los sucesos de temporización lenta. Después se trae de nuevo a la acción a los registros alternativos BC′, DE′ y HL′. Con POP HL cargamos H con el valor que antes contenía A, después, mediante C′=C′ AND &F3 OR H, formamos el valor correcto, que es utilizado para restaurar el estado de la ROM, tal y como estaba antes de la interrupción.

Por un camino o por otro, llegamos finalmente a B96A. Restauramos entonces el estado previo de la ROM mediante OUT (C′),C′. Reseleccionamos los registros normales BC, DE y HL, y recuperamos AF de la pila. Las interrupciones quedan habilitadas y la rutina regresa, reanudándose así la acción de la rutina interrumpida.

Pero, ¿qué hay de esa acción especial que tiene lugar si el acarreo C′ se encuentra a uno? La rutina en B970 parece un poco extraña:

```
B970    EX     AF,AF′    Cancela el cambio anterior
        POP    HL        HL′, para ser exactos
```

```
PUSH  AF
SET   2,C'
OUT   (C'),C'   Inhibe la ROM inferior
CALL  003B      Entrada para interrupciones del usuario
JP    B94B      Sigue la llamada 00B1.
```

Los registros alternativos están en uso, habiendo sido seleccionados por el controlador principal. POP HL' quita la dirección de retorno para la segunda interrupción, como si no fuera necesaria. PUSH AF guarda el par AF normal. Después de llamar a la dirección 003B de la RAM, la rutina principal es introducida inmediatamente después de la llamada a 00B1.

Un controlador empleado por el usuario no debe hacer uso de EXX o EX AF,AF', ya que los registros principales que no se usan están guardando datos del programa interrumpido. Respecto a los registros en uso activo, HL' ya ha sido alterado, mientras que el contenido de BC' debe ser preservado. DE' no parece contener datos de importancia, pero, si se utilizan IX o IY, deberían ser guardados primero en la pila.

Para acciones complicadas, existe una sugerencia oficial que dice que los controladores del usuario deberían estar "anidados", llamándose uno a otro si la interrupción no es asunto propio de cada uno. Esto permite a las unidades externas actualizar sus controladores en el orden que ellos quieran; sin embargo, si se llega a este punto de complejidad, el control de la situación puede escaparse de las manos a cualquiera.

Un requisito específico e importante es que el manipulador de interrupciones debe saber aclarar el motivo y la fuente de la interrupción.

EL sistema de sucesos

La clave del sistema de sucesos es el bloque de sucesos, el cual presenta el siguiente formato:

Bytes 0, 1	Cadena de enlace
Byte 2	Cuenta
Byte 3	Clase
Bytes 4, 5	Dirección de la rutina
Byte 6	Número de la rutina
Bytes 7 en adelante	Campo de usuario.

La cadena de enlace se utiliza para combinar un número de bloques de sucesos en una lista, haciendo que cada enlace apunte al siguiente bloque de suceso. El último bloque de la lista tiene el byte 1 con el valor cero. Esto se estudiará después con más detalle.

"Cuenta" es un archivo de los requisitos más relevantes para la ejecu-

ción. En este sentido, puede contener cualquier valor entre 0 y 127. Cuando un suceso no es atendido, cuenta es incrementada (pero no más allá de 127), y cuando se ejecuta la rutina, cuenta es decrementada. Sin embargo, si cuenta tiene un valor negativo, el suceso se inhibe y cuenta permanecerá inalterada por ejecuciones o por inatenciones.

Clase define el tipo de suceso. El código empleado es:

Bit 0	0; la rutina puede alcanzarse por una "dirección cercana"; esto es, por un simple salto.
	1; se trata de una "dirección lejana" que incluye la selección de una ROM.
Bits 1-4	Prioridad (únicamente para sucesos síncronos)
Bit 5	0
Bit 6	0; suceso normal
	1; suceso urgente
Bit 7	0; suceso síncrono
	1; suceso asíncrono

La dirección de entrada de la rutina y el número de ROM nos señala el acceso a la rutina que se debe llamar para implementar el suceso. El número de ROM será ignorado si se especifica "dirección cercana".

Para sucesos normales, el paso se marca incrementando Cuenta, pero los sucesos urgentes son ejecutados inmediatamente. Es importante que éstos sean lo más breves posible. Los sucesos asíncronos son ligados a interrupciones, mientras que los síncronos son llamados desde el programa principal.

El bloque de cada suceso debe estar contenido en la RAM; de esta forma, su contenido puede ser ajustado. Además, debe estar en la parte central de la RAM, de manera que sea accesible siempre que se necesite.

El campo de usuario que comienza a partir del byte 7 se debe emplear para contener todos aquellos parámetros que sean importantes para la función del suceso.

Un bloque de un suceso se inicializa con esta rutina:

NC INICIALIZA SUCESO *(KL INIT EVENT):* BCEF, 01D2

Al comienzo, HL debe contener la dirección en donde se va a inicializar el bloque del suceso. DE debe contener la dirección de entrada de la rutina asociada, B contendrá el byte de clase y C debe contener el número de la ROM que contiene a la rutina asociada.

Enlace de cadena no se inicializa en esta etapa, y Cuenta se pone a cero. El resto de las entradas se inicializan a partir de los datos dados.

Es aconsejable mantener escritas en una nota las direcciones de los bloques de sucesos, ya que éstos no son fáciles de localizar una vez que han sido inicializados.

Una vez que el bloque ha sido establecido, se deberá enlazar con el resto del sistema. Esto se puede conseguir por diferentes caminos:

NC SUCESO *(KL EVENT):* BCF2, 01E2

Si se llama a NC SUCESO con HL conteniendo la dirección del bloque del suceso, el suceso será "inatendido", dependiendo del resultado de la comprobación de su estado.

Si cuenta es negativo, la rutina regresará, sin que se lleve ninguna acción a cabo. Si cuenta está en el rango 0-126, se incrementa, pero si es 127, la rutina regresará; ya hay demasiadas peticiones importantes. (Como ayuda al diagnóstico, un caso de cuenta = 127 es una clara indicación de que el sistema de interrupciones está sobrecargado.)

Si la rutina no ha regresado, se chequea la clase. Un suceso síncrono se enlaza en la lista de síncronos; un suceso normal será añadido a la lista de inatendidos, y un suceso urgente será implementado inmediatamente.

Los sucesos de la lista de "inatendidos" serán ejecutados en la siguiente interrupción de temporización rápida. Esta lista se construye haciendo que cada cadena de enlace apunte al siguiente bloque de sucesos; el primer enlace estará contenido en (B100/1). Para la lista de sucesos síncronos, el primer enlace estará en (B193/4). Si el byte superior del primer enlace está a cero, la lista está vacía.

Existen otras listas de sucesos asociados con la temporización rápida, con la lenta, o con las interrupciones del barrido de pantalla. Las funciones más importantes son:

NC NUEVO SINCRONO BARRIDO *(KL NEW FRAME FLY):* BCD7, 0163

A su entrada, HL debe contener la dirección en donde se va a inicializar el nuevo bloque asociado al barrido de pantalla. Este consiste en un bloque de sucesos precedido de dos bytes utilizados como cadena de enlace. No se emplea la cadena de enlace habitual de los bloques de sucesos. Los otros parámetros requeridos para NC INICIALIZA SUCESO también se aplican aquí. Primero se crea el bloque del suceso y después se enlaza con la lista de sucesos asociados al barrido de pantalla.

El primer enlace de esta lista está contenido en (B18C/D).

NC AÑADE SINCRONO BARRIDO *(KL ADD FRAME FLY):* BCDA, 016A

A su entrada, HL debe contener la dirección de un bloque de sucesos ya existente, menos dos. El suceso en cuestión será añadido a la lista de sucesos asociados al barrido de pantalla.

NC BORRA SINCRONO BARRIDO *(KL DEL FRAME FLY):* BCDD, 0170

A su entrada, HL debe contener la dirección de un bloque menos dos. El suceso será eliminado de la lista si éste está en primer lugar.

Una vez que un suceso ha sido enlazado en una lista de interrupción, será inatendido cada vez que ocurra dicha interrupción.

Las tres llamadas relativas a las interrupciones de temporización rápida son directamente análogas a las interrupciones asociadas al barrido de pantalla.

NC NUEVO SINCRONO RAPIDO *(KL NEW FAST TICKER):* BCE0, 0176

NC AÑADE SINCRONO RAPIDO *(KL ADD FAST TICKER):* BCE3, 017D

NC BORRA SINCRONO RAPIDO *(KL DEL FAST TICKER):* BCE6, 0183

El primer enlace de la lista de temporización rápida está contenido en (B18E/F).

Los bloques de temporización lenta son más complejos, requiriendo seis bytes que anteceden a un bloque normal de sucesos:

Bytes 0, 1	Cadena de enlace de la temporización lenta
Bytes 2, 3	Cuenta de la temporización
Bytes 4, 5	Recarga de cuenta.

La interrupción de temporización lenta ocurre nominalmente 50 veces por segundo. La cuenta de temporización es entonces decrementada, pero ninguna acción se llevará a cabo hasta que la cuenta valga 1. Cuando esto ocurra, se ejecutará el suceso asociado y la cuenta volverá a inicarse con el contenido de recarga de cuenta. Si recarga de cuenta es cero, el suceso sólo se atenderá una vez, ya que, si cuenta es cero, el suceso queda inhibido. Por otra parte, el suceso puede llamarse a intervalos, incluso algo superiores a los 21 minutos.

El primer enlace de la lista de temporización lenta está contenido en (B190/1).

NC AÑADE SINCRONO LENTO *(KL ADD TICKER):* BCE9, 01B3

A su entrada, HL debe contener la dirección del bloque del suceso menos seis, DE debe contener la cuenta inicial y BC contendrá la recarga de cuenta. El suceso se añade a la lista de temporización lenta (no existe ninguna función para crear un suceso y añadirlo a la lista a la vez).

NC BORRA SINCRONO LENTO *(KL DEL TICKER):* BCEC, 01C5

A su entrada, HL debe contener la dirección del bloque del suceso menos seis. El suceso se elimina de la lista de temporización lenta.

Observa que la eliminación de un suceso de una lista deja intacto al bloque del suceso y que éste puede ser devuelto a la lista posteriormente si fuera necesario.

NC IGNORA SUCESO *(KL DISARM EVENT):* BD0A, 028E

Al comienzo, HL debe contener la dirección de un bloque de sucesos. El byte de cuenta del bloque se hace negativo, de forma que el suceso queda desactivado indefinidamente.

Sucesos síncronos

Los sucesos síncronos se manipulan de manera algo diferente: los sucesos se disponen en la lista de síncronos según su orden de prioridad. El primer enlace de la lista está contenido en (B193/4) y el nivel actual de prioridad se almacena en (B195).

NC REINICIALIZA SINCRONOS *(KL SYNC RESET)*: BCF5, 0228

Con esta rutina se pone a cero el contenido de (B194/5) indicando que la lista está vacía y poniendo al cero como nivel de prioridad.

NC BORRA SUCESO *(KL DEL SYNCHRONOUS)*: BCF8, 0285

A su entrada, HL debe contener la dirección de un bloque de sucesos, que será eliminado de la lista de sucesos síncronos. Se llama a NC IGNORA SUCESO para hacer negativo el byte cuenta y que la cadena de enlace que apunta hacia este bloque pase a apuntar al siguiente suceso de la lista. Si no hubiese ningún suceso después del eliminado, el byte superior del enlace se pondrá a cero.

Para los sucesos síncronos no-urgentes, los bits 5-7 del byte "clase" están a cero; de esta forma, el valor de clase depende sólo de los bits de prioridad 1-4 y del bit de tipo de dirección, bit 0. Cuando NC SUCESO encuentre que se trata de un suceso síncrono, la subrutina de atención a síncronos será llamada inmediatamente. Esta explorará la lista de sucesos síncronos hasta alcanzar el final o hasta que el byte de clase de un suceso listado sea menor que el byte de clase del nuevo suceso, lo cual significa que el nuevo suceso tiene mayor prioridad.

Sea el suceso N de la lista, el suceso anterior, N-1, tendrá a su cadena de enlace apuntando al bloque del suceso N. Esta se modifica de forma que apunte hacia el nuevo suceso, mientras que la cadena de enlace del nuevo suceso toma el valor que antes contenía la cadena de enlace del suceso N-1. El bloque del nuevo suceso se inserta de este modo en la lista, atendiendo a su prioridad.

NC BORRA SUCESO realiza este proceso cambiando los enlaces, de forma que se evite el bloque del suceso borrado.

NC SIGUIENTE SUCESO *(KL NEXT SYNC):* BCFB, 0256

Esta función revisa la lista de sucesos síncronos en busca de alguna, con una prioridad mayor que la que contiene (B195). Si no se encuentra ningún suceso así, la rutina regresará con el acarreo a cero.

Si se identifica algún suceso adecuado, la rutina regresará con el acarreo a uno, HL contiene la dirección del bloque del suceso y A contendrá la prioridad del suceso (clase), que también se almacena en (B195). El suceso se quita de la lista de síncronos.

Cuando NC SIGUIENTE SUCESO encuentra un suceso apropiado, éste se procesa por:

NC EJECUTA SUCESO *(KL DO SYNC):* BCFE, 021A

A su entrada, HL debe apuntar a un bloque de sucesos, como el localizado por SIGUIENTE SUCESO. Se llama a la rutina del suceso y se ejecuta. Para completar la acción es necesario llamar a:

NC SUCESO EJECUTADO *(KL DONE SYNC):* BD01, 0277

A su entrada, HL debe apuntar al bloque de sucesos relevante. La dirección no proviene de EJECUTA SUCESO, por lo que la dirección obtenida con SIGUIENTE SUCESO deberá guardarse en la pila mientras se está ejecutando EJECUTA SUCESO. Al igual que antes, A debe contener la prioridad del suceso anterior, dato ofrecido también POR SIGUIENTE SUCESO y no por EJECUTA SUCESO. La documentación oficial especifica C en vez de A, pero el código máquina del aparato utiliza A en realidad. El nivel de prioridad en (B195) se guarda desde A y se decrementa la cuenta del bloque del suceso. Si la cuenta es positiva y distinta de cero, el suceso es devuelto a la lista de síncronos.

NC PRIORIDAD *(KL POLL SYNCHRONOUS):* B921

Esta es una rutina de RAM a la que se accede directamente para hacer una rápida comprobación del primer suceso de la lista de síncronos. Si éste tiene una prioridad mayor que la prioridad indicada por (B195), la rutina regresa con el acarreo puesto a uno.

NC INHIBE SUCESOS *(KL EVENT DISABLE):* BD04, 0295

El bit 5 de (B195) se pone a uno, indicando esto que es imposible una prioridad mayor. Por tanto, ningún suceso será atendido.

NC DESINHIBE SUCESOS *(KL EVENT ENABLE):* BD07, 029B

Se pone a cero el bit 5 de (B195).

Las anteriores descripciones son perfectamente correctas, pero dejan sin responder algunas cuestiones. Por ejemplo, ¿cómo puede una llamada fijar el valor de (B195)? ¿Se necesita algo para hacerlo?

Veamos el sistema desde una perspectiva más amplia. La intención es que el programa primario realice comprobaciones regulares de los sucesos síncronos relevantes. Esto puede conseguirse llamando a NC PRIORIDAD. Si el regreso se efectúa con el acarreo a uno, la secuencia

```
L1    CALL    NC, SIGUIENTE SUCESO
      JR      NC, SALIDA
      PUSH    HL
      PUSH    AF
      CALL    NC EJECUTA SUCESO
      POP     AF
      POP     HL
      CALL    NC SUCESO EJECUTADO
      JP      L1
```

empezará a correr. Esto procesará todos los sucesos en su nivel de prioridad. El nivel será 0 inicialmente, ya que con la inicialización (B195) toma el valor cero, pero, tan pronto como NC SIGUIENTE SUCESO encuentra un suceso con máxima prioridad, la prioridad actual se establece a ese nivel, y todos los sucesos de prioridad inferior son arrastrados a continuación.

Lo que aquí se necesita es una extensión de la rutina anterior. Cuando se alcance la SALIDA, si (B195) no es cero, entonces se decrementa y la rutina vuelve a comenzar. Para aquellos a los que no les agrade la idea de comprobar si (B195) es la posición correcta de su versión del sistema, diré que es posible cargar (B195) llamando a NC SUCESO EJECUTADO con el valor requerido en A y con HL apuntando a un bloque de sucesos "postizo" y únicamente válido para esta comprobación.

Comentarios

El sistema de sucesos es un diseño que dista mucho de ser como algo que se pueda completar en una sola tarde de trabajo. Como las longitudes de las diferentes listas son virtualmente ilimitadas, es posible perder las riendas y crear demasiados sucesos; tantos, que el sistema no tuviera tiempo de atender a ninguno más. Es recomendable, por tanto, un poco de precaución.

Se pueden escribir programas ciertamente complejos sin hacer uso de los sucesos, pero, una vez que se comprende el sistema de interrupciones, aparece un amplio horizonte para el ingenio.

Debido a que el contenido de las listas cambia rápidamente durante todo el tiempo, puede presentar bastante dificultad describir de antemano qué va a pasar o qué está pasando. Los sucesos nos proporcionan una poderosa herramienta, pero, igual que todas las herramientas poderosas, es necesario manejarla con cuidado.

Otras rutinas del núcleo

Algunas de las rutinas del núcleo no son accesibles a través del bloque de saltos. Hemos visto que en 0077 está la rutina de entrada a los programas; en 0044 hay una rutina que copia en la RAM código de la ROM. También hay otras rutinas para añadir un suceso a una lista, o borrar un suceso, pero no es conveniente utilizarlas de forma aislada. Dos rutinas más del núcleo, accesibles a través del bloque de saltos, pueden bastar aquí:

NC TIEMPO *(KL TIME PLEASE):* BD0D, 0099

El contenido del contador de tiempo se transfiere a DE (2 bytes superiores) y HL (2 bytes inferiores).

NC PONE TIEMPO *(KL TIME SET):* BD10, 00A3

El contador de tiempo se carga desde DEHL, con (B18B)=0.

La compacta rutina empleada para incrementar el contador de tiempo se describe aquí de forma funcional:

```
L1        HL=B187
```

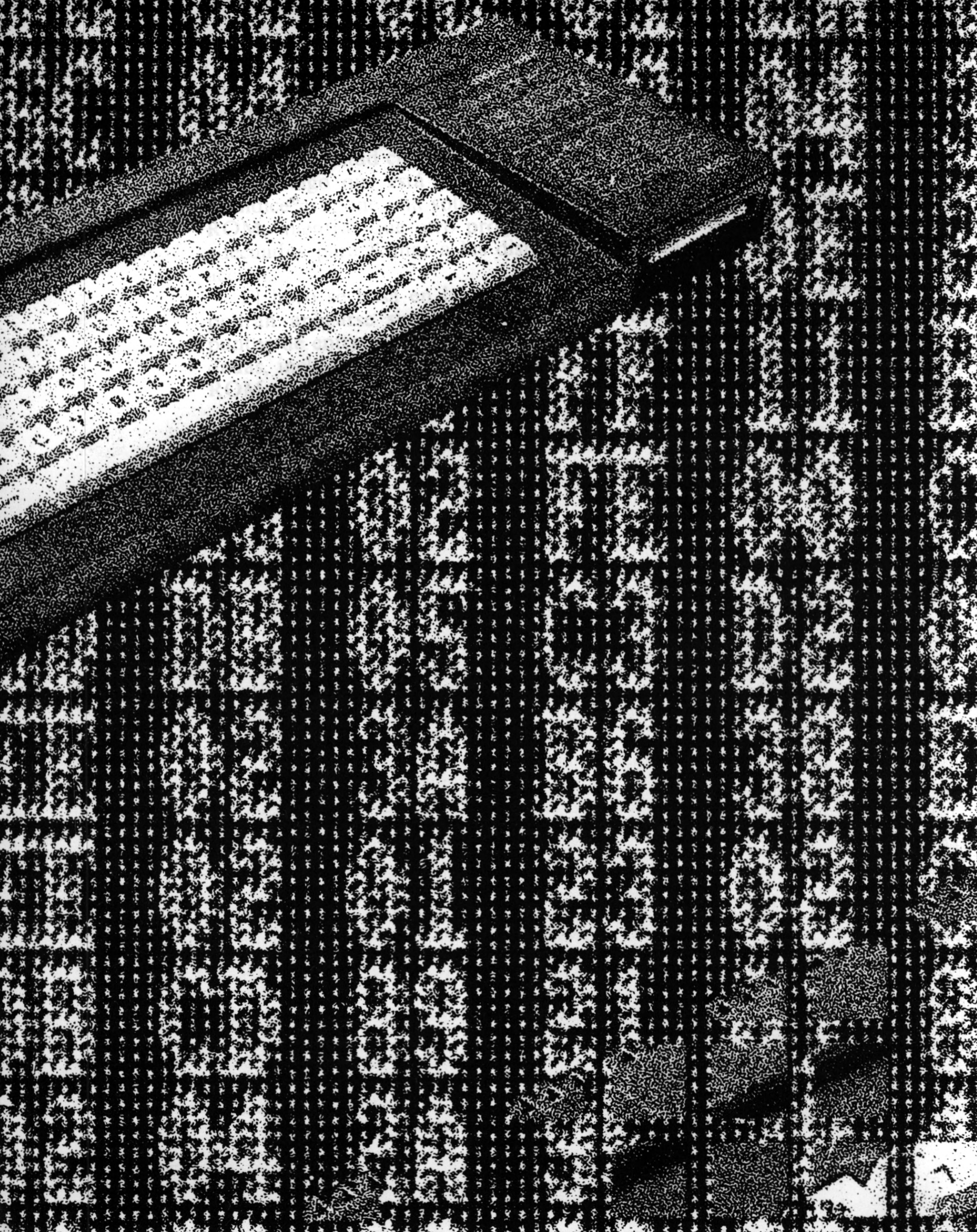
```
L2      INC (HL)
        INC HL
        IF HL=0 THEN L2
        RET
```

Si un byte se incrementa desde &FF a 0, el acarreo se traslada al siguiente byte. Esto puede inducir a un sobrevuelo en (B18B), cuando FFFF FFFF se incremente. Aproximadamente esto ocurre una vez cada 4.000 horas, pero (B18B) no volverá a ser cero hasta que transcurra un millón de horas. Puede que el sistema falle entonces, si eres capaz de esperar tanto tiempo.

Hay cuatro rutinas más del núcleo que están muy enlazadas con el sistema de ROM externas, que son las que mejor trato tienen con este contexto.

Area de datos del núcleo

B100-B101	Base de la lista de inatendidos
B102-B103	Fin de la lista de inatendidos
B104	Byte de *flag*
B105-B106	Contiene el SP (puntero de pila)
B107-B186	Pila especial
B187-B18B	Contador de tiempo
B18C-B18D	Base de lista de sucesos asociados barrido
B18E-B18F	Base de lista de temporización rápida
B190-B191	Base de lista de temporización lenta
B192	Contador de temporización lenta
B193-B194	Base de lista de síncronos
B195	Prioridad actual
B196-B1A5	Copia de las palabras-órdenes
B1A6-B1A7	Base de la cadena de órdenes
B1A8	ROM actual
B1A9-B1AB	Dirección lejana característica
B1AC-B1B8	Punteros del área de datos.



5
Sistema de pantalla

En total, el sistema de pantalla utiliza unos 4000 bytes de códigos y datos fijos en la ROM, y su espacio de trabajo ocupa unos 380 bytes de la RAM, sin mencionar los 16 Kbytes de la RAM de pantalla. Hay más de 100 puntos de entrada, pero afortunadamente el sistema se divide en tres partes fundamentales:

- El grupo de pantalla, que se ocupa directamente del manejo de la pantalla, selección de color, lectura y escritura en pantalla.
- La VDU de texto trata las materias relativas a salida de textos, incluyendo la implementación de la selección de cauces. También se ocupa de los códigos de control y sus parámetros.
- La VDU de gráficos se ocupa de la salida de gráficos en la pantalla.

Cada una de estas partes requiere un capítulo individual, pero es conveniente tener cierta información general previamente.

La RAM de pantalla

En teoría, la RAM de pantalla podría ser cualquier área de la memoria de 16 Kbytes que comenzara en un múltiplo de 4000; pero el bloque 8000-BFFF superpondría el espacio de trabajo y las rutinas de la RAM,

mientras que 0000-3FFF haría lo mismo con el área RST; así que la elección queda reducida al campo 4000-7FFF o C000-FFFF, y normalmente es más conveniente escoger esta última área, quedando la mitad central de la RAM libre para otras funciones.

La RAM de pantalla es accesible mediante la matriz de puertos de video, a base de direcciones proporcionadas por el controlador del CRT; pero las direcciones no son manejables directamente. El controlador del CRT incorpora dos contadores. Uno, cuya salida está en RA0-RA4, se incrementa cuando cada línea de la pantalla haya sido barrida por el haz. Cuando esta cuenta alcance el valor asignado para el número de líneas de la altura del carácter, se pondrá a cero, y el segundo contador, cuya salida está en MA0-MA13, se incrementa. Este contador se inicializa en la dirección de comienzo, dispuesta en el controlador del CRT, que será 3000 cuando el área C000-FFFF esté utilizándose. Estas salidas se utilizan del siguiente modo:

- Los bits A14, A15 de la dirección son dirigidos desde MA12, MA13. Puesto que el contador MA trabaja desde una dirección de comienzo de 3000, ambos bits están puestos a 1.
- Los bits A11-A13 de la dirección se dirigen desde RA0-RA3.
- Los bits A1-A10 de la dirección se dirigen desde MA0-MA9.
- El bit A0 de la dirección se controla desde el reloj del controlador del CRT.

La línea del haz tarda 40 microsegundos en atravesar la parte visible de la pantalla, y durante cada microsegundo la matriz de puertas de video toma dos bytes de datos de pantalla. Estos se transfieren directamente desde la RAM a la matriz de puertas de video, mientras que el procesador se mantiene en espera. El proceso está cronometrado, de forma que el reloj del controlador del CRT modifica su estado entre las dos transferencias. Una vez hayan sido leídos todos los bytes, puede continuar la acción normal del procesador.

Los bytes se utilizan de forma distinta en los tres modos de pantalla.

En modo 2, cada byte define una fila de una matriz modelo de un carácter, determinando cada bit cual de los dos colores debe asignarse a un punto, y 80 caracteres se dibujan en cada fila de la pantalla.

En modo 1, se requieren dos bytes para definir cada fila de la matriz, utilizándose dos bits para asignar a cada punto uno de los cuatro colores posibles. Los puntos sucesivos se definen mediante los bits 3,7; 2,6; 1,5, y 0,4. Esta secuencia se repite en el segundo byte. A partir de cada pareja de bits, la matriz de puertas de video determina qué entrada de la paleta debe utilizarse y dispone el color correspondiente. Puesto que cada fila de matriz requiere dos bytes, sólo pueden dibujarse 40 caracteres por fila de pantalla.

En modo 0, se requieren cuatro bits para definir uno de los dieciséis colores asignables a cada punto. Es decir, que se necesitan cuatro bytes

para cada fila de matriz. El primer punto se define mediante los bits 1, 5, 3, 7 del primer byte; el segundo, mediante los bits 0, 4, 2, 6, y así sucesivamente. Pueden dibujarse veinte caracteres en cada fila de pantalla.

El modo en que se utilizan los contadores del controlador del CRT complica el cálculo de las direcciones de pantalla. Numerando filas y columnas de 0 en adelante:

Dirección = Base + Offset + N*Columna + 80*Fila + 2048 por línea del haz,
donde N es el número de bits por punto en el modo actual.

Para una línea dada del haz, los bits se toman en secuencia. La siguiente línea del haz se localiza incrementando las direcciones en 0800.

Afortunadamente, el sistema calculará las direcciones de pantalla a partir de la columna, línea, base y *offset*.

Si sois observadores, habréis advertido una pequeña anomalía. Si N*Columna = 79 y Fila = 25, los términos columna y fila en la ecuación anterior totalizarán 1999, de forma que habrá 48 posiciones de más en línea del haz. El contador MA del controlador del CRT no las direcciona.

Sin embargo, el término *offset* debe añadirse a la cuenta. Si el offset = &50, conseguimos que la pantalla se desplace una línea hacia arriba. Si es igual a 800, se desplazará el dibujo una línea más arriba, pero el *offset* no puede exceder de 07FF, porque lo limita la rutina utilizada normalmente para especificarlo. Cuando se utiliza el *offset*, la dirección de comienzo guardada en el controlador del CTR se modifica y los 48 bytes sobrantes se pueden utilizar entonces. Hay mucho campo aquí para los experimentadores.

Cauces y ventanas

Para la definición de los datos de pantalla, el sistema dispone de ocho cauces *(streams)*, cada uno con sus parámetros independientes, que son:

Ventana
Posición del cursor
Pluma y papel
Activación del cursor
Activación de la pantalla
Modo opaco o transparente
Impresión de texto o gráficos
Tipo de desplazamiento

Estos ocho grupos de parámetros se mantienen almacenados, y el grupo que se esté utilizando se copia en un área común.

Cada cauce puede reservar para sí una ventana rectangular. Si dos ventanas se solapan, los cauces pueden superponerse en esa área coincidente. Las zonas no reservadas son accesibles mediante el cauce 0, que es el prescrito para omisiones.

Parámetros

Hay que ser bastante cauto a la hora de manejar los parámetros de pantalla, puesto que sus definiciones son susceptibles de variación. Se distingue entre valores "físicos" y "lógicos", numerando los primeros columnas y filas de 0 en adelante, mientras que los segundos lo hacen a partir de 1. También existen diferencias entre valores absolutos y relativos.

Surgen distinciones análogas para los parámetros de gráficos, al considerarse las coordenadas del usuario relativas respecto del origen determinado por el propio usuario, mientras que las coordenadas estándar son relativas respecto del origen natural del aparato.

El espacio de trabajo

Dado que muchas posiciones del espacio de trabajo son comunes a más de una sección del sistema de pantalla, se proporcionan aquí las direcciones para todo el espacio de trabajo de la pantalla para la versión 1.0.

Compresión de pantalla

B1C8	Modo
B1C9-B1CA	Offset
B1CB	Base (byte alto)
B1CC-B1CE	Instrucción de saltos
B1CF-B1D6	Máscaras de puntos
B1D7	Segundo tiempo de parpadeo
B1D8	Primer tiempo de parpadeo
B1D9-B1E9	Tabla 2 de colores
B1EA-B1FA	Tabla 1 de colores
B1FB	*Flag* de selección de tabla
B1FC	Contador de parpadeos
B1FD	Tiempo de cada color
B1FE-B206	Bloque de sucesos
B207	Bits que forman el punto, negados.

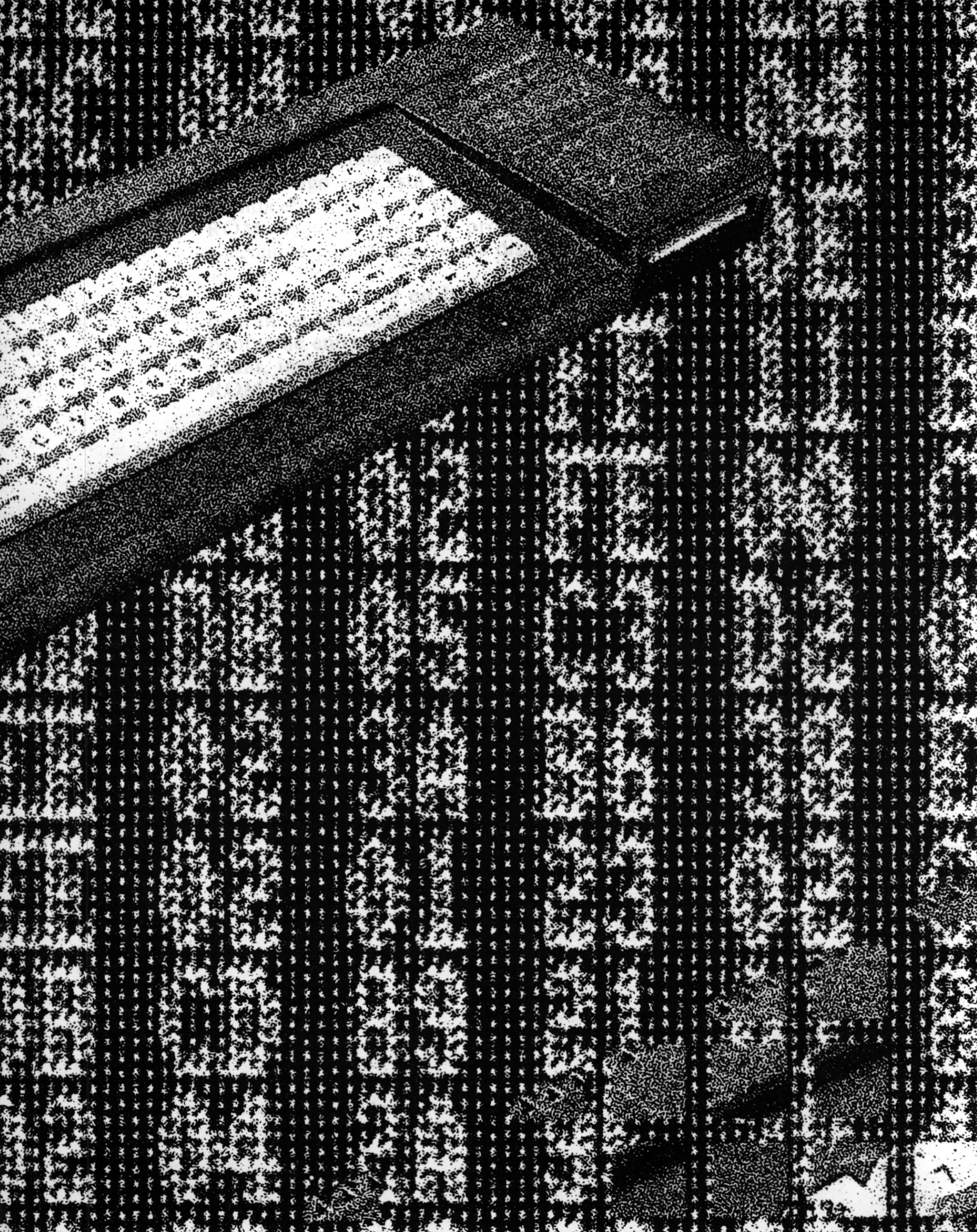
Gráficos

B328-B329	Coordenada X del origen
B32A-B32B	Coordenada Y del origen
B32C-B32D	Posición X
B32E-B32F	Posición Y
B330-B331	Izquierda de la ventana
B332-B333	Derecha de la ventana
B334-B335	Borde superior de la ventana
B336-B337	Borde inferior de la ventana
B338	Pluma codificada
B339	Papel codificado
B33A-B341	Copia de matriz
B342-B343	Coordenada X elegida
B344-B345	Coordenada Y elegida

VDU de texto

B20C	Cauce actual
B20D-B21B	Datos del cauce 0
B21C-B22A	Datos del cauce 1
B22B-B239	Datos del cauce 2
B23A-B248	Datos del cauce 3
B249-B257	Datos del cauce 4
B258-B266	Datos del cauce 5
B267-B275	Datos del cauce 6
B276-B284	Datos del cauce 7
B285	Fila actual
B286	Columna actual
B287	Tipo de desplazamiento actual
B288	Borde superior actual
B289	Borde izquierdo actual
B28A	Borde inferior actual
B28B	Borde derecho actual
B28C	Cuenta de desplazamiento actual
B28D	*Flag* del cursor actual
B28E	*Flag* de activación de la pantalla actual
B28F	Pluma actual
B290	Papel actual
B291-B292	Conexión para modo de impresión
B293	*Flag* de escritura de gráficos
B294	Código de la primera matriz de la RAM
B295	*Flag* de la matriz

B296-B297	Dirección de la matriz de la RAM
B298-B2B7	Modelo elegido
B2B8	Contador de parámetros
B2B9	Código de control
B2BA-B2C2	Parámetros de control
B2C3-B322	Tabla de saltos de control



6
El control de pantalla

Las rutinas de control de pantalla ocupan el área 0AA0-106E de la ROM, y se compone de 34 puntos de entrada definidos y de tres indirecciones. Las rutinas se seleccionan con la selección del modo de pantalla, cálculo de direcciones, control del color y asuntos similares. Comenzaremos viendo las rutinas de inicialización.

PNT INICIALIZA *(SCR INITIALISE):* BBFF, 0AA0

Se llama a MC BORRA TINTAS con DE=104D, limpiando todas las entradas de la paleta de colores y poniéndolas a &04. La base de pantalla se fija en C000 y a continuación se llama a PNT RE-INICIALIZA y PNT BORRA PANTALLA.

PNT RE-INICIALIZA *(SCR RESET):* BC02, 0AB1

Se llama a PNT MODO GRAFICOS con A=0 para seleccionar el modo normal de escritura. Las indirecciones PNT LEE PIXEL, ESCRIBE PIXEL y LIMPIA MODO se reinicializan con las direcciones que tienen

asignadas normalmente. Se llama a 0CD2 para copiar los datos de color desde 104D-106E en las tablas de dos colores situadas en B1D9-B1E9 y B1EA-B1FA. Los tiempos de parpadeo se fijan a un quinto de segundo. El *flag* de selección de color (B1FB) se pone a cero.

PNT BORRA PANTALLA *(SCR CLEAR):* BC14, 0AF2

Se selecciona el modo 1, con la máscara apropiada puesta. A continuación se llama a LIMPIA MODO.

PNT LIMPIA MODO *(SCR MODE CLEAR):* BDEB, 0AF7

Esta es una indirección y no deberá llamarse cuando la ROM inferior esté inhibida. Se llama a 0D4F para inhibir el sistema de parpadeo, que será examinado posteriormente. Se llama a PNT IMPONE OFFSET con HL=0000 para estandarizar el mapa de pantalla, después se pone a cero toda la RAM de pantalla por medio de la instrucción LDIR. La rutina termina a través de 0D3C para reactivar el sistema de parpadeo.

Control del modo

PNT FIJA MODO *(SCR SET MODE):* BC0E, 0ACA

Al acceder a ella, A debe contener el número del modo requerido. Si el número está fuera del rango 0-2, la rutina regresa inmediatamente.

En caso correcto, se llama a 0D4F para inhibir el sistema de parpadeo, después se llama a 10B7 para inicializar las ventanas *(streams)*, siendo ésta una rutina que pertenece al área de la VDU de texto. A continuación se llama 15D6 de la VDU de gráficos para fijar el papel y la pluma de gráficos. La tabla de máscaras se inicializa entonces como se indica debajo.

El número del modo se guarda en (B1CB) y se llama a MC MODO PANTALLA para reinicializar la matriz de puertas de video. Seguidamente se efectúan llamadas a PNT LIMPIA MODO, GRA INICIALIZA en 15B6 y GRA RE-INICIALIZA. La salida final se hace vía 10D5, donde está TXT SELECCIONA CAUCE.

Tabla de máscaras

	Modo 0	Modo 1	Modo 2
B1CF	&AA	&88	&80
B1D0	&55	&44	&40
B1D1	*	&22	&20
B1D2	*	&11	&10
B1D3	*	*	&08
B1D4	*	*	&04
B1D5	*	*	&02
B1D6	*	*	&01

Los asteriscos indican entradas que son inicializadas pero no utilizadas.

La tabla toma los bits de los bytes de datos de la pantalla que son empleados para definir los *pixels* individuales. Por ejemplo, en el modo 1 el primer *pixel* se define por los bits 3 y 7, luego la máscara será &88.

PNT EXAMINA MODO *(SCR GET MODE):* BC11, 0AEC

A se carga con el contenido de (B1C8) y se compara con 1. Esto hará que los *flags* tomen diferentes valores: C a uno y Z a cero para el modo 2. Existe un cierto número de llamadas internas a esta rutina, y se utiliza más a menudo el estado de los *flags* que el número contenido en A.

Direcciones

PNT IMPONE OFFSET *(SCR SET OFFSET):* BC05, 0B3C

A su entrada, HL debe contener el desplazamiento de pantalla requerido. H=H AND 7 y después (B1C9/A)=HL. Se llama a PNT LOCALIZA RAM PANTALLA y la rutina finaliza a través de MC OFFSET PANTALLA, que reinicializa el controlador del CRT.

Oficialmente, el desplazamiento (OFFSET) dado está limitado a un número en el rango 0-07FE, lo cual es deseable, ya que números incorrectos pueden provocar confusión, aunque la rutina no haga cero al bit 0. El usuario debe atender a esta limitación.

PNT IMPONE BASE RAM *(SCR SET BASE):* BC08, 0B45

El byte superior de la base requerida de pantalla debe estar contenido en A al acceder a la rutina. Este se enmascara con A=A AND &C0, permitiendo que la base sea 0000, 4000, 8000 o C000, y el resultado se guarda en (B1CB). Se llama entonces a PNT LOCALIZA RAM PANTALLA y la rutina termina con una llamada a MC OFFSET PANTALLA para reinicializar al controlador del CRT.

PNT LOCALIZA RAM PANTALLA *(SCR GET LOCATION):* BC0B, 0B50

HL=(B1C9/A), que es el offfset, y A=(BC1B), que es el byte superior de base.

PNT LIMITES PANTALLA *(SCR CHAR LIMITS):* BC17, 0B57

Se llama a PNT EXAMINA MODO. Para todos los modos C=&18 (líneas de pantalla menos uno), mientras que B se carga con el número de columnas de pantalla menos uno.

Estos valores marcan las "coordenadas físicas", que comienzan en 0.

PNT POSICIONA COORDENADAS *(SCR CHAR POSITION):* BC1A, 0B64

A la entrada, H debe contener el número de la columna física y L, el número de la fila física. La dirección de pantalla correspondiente se calcula y se devuelve en HL. El número de bits por *pixel* para el modo actual de pantalla se introduce en B.

PNT COORDENADA PUNTOS *(SCR DOT POSITION):* BC1D, 0B95

En realidad, ésta es una función de gráficos. A su entrada, DE debe contener la coordenada X de un *pixel* y HL debe contener la coordenada Y, estando ambas expresadas en términos de desplazamiento absoluto

desde la esquina inferior izquierda de la pantalla. La dirección de pantalla del byte relativo al *pixel* se devuelve en HL, mientras que B contiene el número de bits por *pixel* menos uno y C contiene una máscara que identifica los bits relevantes del byte especificado de la pantalla.

Llegamos ahora a cuatro rutinas que no son de ninguna manera fáciles de seguir. En cada caso se modificará HL para que apunte a un byte en una posición adyacente de la pantalla.

PNT SIGUIENTE BYTE *(SCR NEXT BYTE):* BC20, OBF9

L es incrementado y, si el resultado es distinto de cero, la rutina regresa. En otro caso, se requiere un acarreo de H, por lo que se incrementa H. Si ocurre que H AND 7=0, entonces H=H−&08. Se ha alcanzado el final de un bloque y se requiere una corrección.

PNT BYTE ANTERIOR *(SCR PREV BYTE):* BC23, 0C05

L es decrementado y, si no era previamente igual a cero, la rutina regresa. En otro caso se decrementa H y, si previamente H AND 7≠0, la rutina regresa. De otra forma, H=H+&08 para aplicar la corrección necesaria.

PNT SIGUIENTE LINEA *(SCR NEXT LINE):* BC26, 0C13

H=H+8, moviendo el correspondiente byte a la siguiente línea del haz de pantalla. Si H AND &38≠0, la rutina regresa. En caso contrario, la dirección está fuera del margen y H=H−&40, L=L+&50, llevará la dirección al otro extremo de la RAM de pantalla. Finalmente, si H AND 7≠0, entonces H=H−8.

PNT LINEA ANTERIOR *(SCR PREV LINE):* BC29, 0C2D

H=H−8. Si H AND &38≠&38, la rutina regresa. En otro caso, H=H+&40, L=L−&50. Si H AND 7=0, entonces H=H+8.

Es útil dibujar la RAM de pantalla como dividida en ocho secciones, cada una de las cuales trata con una fila en particular de la matriz de cada carácter. Puede ayudar el dibujar un mapa de parte de la pantalla, pero utiliza una hoja de papel bien grande para ello.

Tintas y parpadeo de colores

El sistema de colores implica unas traslaciones un tanto desconcertantes, pero por otra parte su seguimiento es muy directo.

PNT CODIFICA TINTA *(SCR INK ENCODE):* BC2C, 0C86

El número de tinta contenido en A al comienzo de la rutina se convierte en una máscara para tintas y se devuelve en A.

Primero se llama a 0CC2 para intercambiar los bits 1 y 2 de A si el modo 0 está en uso. Después comienza un bucle de ocho iteraciones con E conteniendo inicialmente el número de tinta, original o modificada, y C contiene (B1CF), que es la primera máscara de color.

El bit 0 de E se copia en el bit 0 de A, siendo rotado E a la derecha y A hacia la izquierda en el mismo proceso. C se desplaza hacia la derecha y, si el bit transferido al acarreo desde C es cero, E rotará a la izquierda, restaurando su contenido anterior. La rutina continúa entonces con el bucle.

Para el modo 2, C contiene &80, por lo que el contenido de E permanecerá inalterado hasta la última iteración. El bit 0 de E se transferirá a todas las posiciones de A, cada una de las cuales está relacionada con un *pixel.*

Para el modo 1, C contiene &88. El bit 0 de E se transfiere a los bits 4 a 7 de A, y el bit 1 de E se transferirá a los bits 0-3 de A.

Para el modo 0, recordemos el intercambio de bits. Los bits de A se dispondrán del siguiente modo:

Bit de A	7	6	5	4	3	2	1	0
Bit de E	0	0	1	1	2	2	3	3
Bit de color	0	0	2	2	1	1	3	3

PNT DECODIFICA TINTA *(SCR INK DECODE):* BC2F, 0CA0

Se invierte el proceso antes descrito, convirtiéndose la tinta codificada contenida en A a la entrada de la rutina en un número de tinta que se devolverá en el registro A.

Las dos anteriores conservan el contenido de BC, DE y HL.

PNT IMPONE TINTA *(SCR SET INK):* BC32, 0CEC

PNT IMPONE BORDE *(SCR SET BORDER):* BC38, 0CF1

Estas dos entradas forman una rutina común. Al acceder a ellas, B y C deben contener números de color, los cuales han de ser iguales para que no haya parpadeo, o diferentes para conseguir el efecto de parpadeo. Para IMPONE TINTA, A debe contener un número de tinta, pero para IMPONE BORDE, A se pone a cero. En el caso de IMPONE TINTA, A=A AND &0F+1 dará el rango entre 1 y &10.

Uno puede esperar que el proceso subsiguiente, común para ambas entradas, sea muy sencillo, siendo introducidos los números de color en las posiciones de las dos tablas de colores indicadas por los números de tinta. Pero, no obstante, se requiere un proceso adicional, debiendo reconvertirse el número de color, según la referencia de la siguiente tabla:

&ØØ	&14	&Ø8	&ØD	&1Ø	&Ø7	&18	&ØA
&Ø1	&Ø4	&Ø9	&16	&11	&ØF	&19	&Ø3
&Ø2	&15	&ØA	&Ø6	&12	&12	&1A	&ØB
&Ø3	&1C	&ØB	&17	&13	&Ø2	&1B	&Ø1
&Ø4	&18	&ØC	&1E	&14	&13	&1C	&Ø8
&Ø5	&1D	&ØD	&ØØ	&15	&1A	&1D	&Ø9
&Ø6	&ØC	&ØE	&1F	&16	&19	&1E	&1Ø
&Ø7	&Ø5	&ØF	&ØE	&17	&1B	&1F	&11

El número de color en la izquierda de cada par representa el número de color que hay a la derecha de cada par. Observa que sólo se utiliza la primera mitad de la tabla. Los números reconvertidos se introducen en las dos tablas de colores y, entonces, (B1FC)=&FF para advertir al sistema de color que los colores se han cambiado.

PNT OBTIENE TINTA *(SCR GET INK):* BC35, 0D14

PNT OBTIENE BORDE *(SCR GET BORDER):* BC3B, 0D19

Aquí también se emplea una rutina común, con la excepción de que OBTIENE TINTA requiere un número de tinta a su entrada contenido en A, mientras que OBTIENE BORDE pone A a cero. La tinta se lee de las tablas de colores, y después se reconvierten con la referencia inversa de la tabla antes indicada. Los resultados se devuelven en B y C, con el primer color en B.

PNT FIJA FLASH *(SCR SET FLASHING):* BC3E, 0CE4

El contenido de HL a la entrada se almacena en (B1D7/8). Los dos bytes indican los períodos de parpadeo del color en cincuentavos de segundo. El tiempo para el primer color viene dado por el segundo byte, mientras que el primer byte fija el tiempo para el segundo color.

PNT EXAMINA FLASH *(SCR GET FLASHING):* BC41, 0CE8

HL=(B1D7/8); véase la rutina anterior.

El sistema de parpadeo

El parpadeo de colores se ejecuta automáticamente por un suceso de la lista de sucesos asociados al barrido de pantalla. Los detalles del suceso son:

Dirección del bloque de sucesos: B200
Clase: Asíncrono, dirección cercana
Dirección de la rutina: 0D5B

La rutina de este suceso pertenece a un grupo de pequeñas rutinas muy entrelazadas. Este grupo se describirá a continuación en orden de dirección.

Rutinas llamadas por PNT borra pantalla

0D3C. El suceso se elimina de la lista de asociados al barrido de pantalla; se llama a 0D6D, y el suceso regresa a la lista.

0D4F. El suceso se elimina de la lista. 0D81 cargará DE y A, finalizando la rutina vía MC ASIGNA TINTAS.

Rutina de sucesos

0D5B. Se decrementa la cuenta actual de parpadeo en (B1FD) y, si el resultado es cero, se llama a 0D6D para cambiar los colores. Si (B1FC)≠0, indicando que los colores han sido reinicializados, se llama a 0D81 para cargar DE y A, llamando a continuación a MC ASIGNA TINTAS. Por último, (B1FC)=0.

LLAMADA POR 0D3C y 0D5B

0D6D. Se llama a 0D81 para cargar DE y A, después (B1FD)=A, fijando la cuenta actual de parpadeo. Se llama a continuación a MC ASIGNA TINTAS, se complementa el *flag* de selección de color en (B1FB), y (B1FC)=0.

LLAMADA POR 0D4F, 0D5B y OD6D

0D81. Se dispone a DE de forma que apunte a una tabla de colores, y A se carga en la cuenta de parpadeo. Si (B1FB)=0, se selecciona el primer color siendo los datos B1EA, (B1D8). En caso contrario, el segundo color está representado por B1D9, (B1D7).

Ninguna de estas rutinas es accesible a través del bloque de saltos.

PNT RELLENA CAJA *(SCR FILL BOX):* BC44, 0DB3

PNT RELLENA BYTE *(SCR FLOOD BOX):* BC47, 0DB7

La diferencia entre estas dos rutinas está en la forma en que se expresan los parámetros de entrada. PNT RELLENA CAJA llama a una rutina de conversión de parámetros en 0B95 y después se ejecuta PNT RELLENA BYTE.

A la entrada de ambas rutinas, A debe contener la tinta a utilizar, ya codificada. Para RELLENA CAJA, H = columna izquierda, L = línea superior, D = columna derecha, E = línea inferior. Estas son coordenadas físicas, desde 0 en adelante.

0B95 calcula E = (E − L + 1)*8, es decir, el número de líneas de barrido que forma la altura de la caja. Después D = (D − H + 1), número de caracteres en el ancho de la caja. HL se preserva. Entonces se llama a PNT POSICIONA COORDENADAS para devolver en HL la dirección de la esquina superior izquierda de la caja (definida en HL). También se utiliza B, para dar la relación bits/*pixel* en el modo actual, permitiendo esto realizar el cálculo D = D + 8, que dará el número de bytes que tiene la caja de ancho. C = A.

Los requisitos de entrada para PNT RELLENA BYTE son precisamente las mismas condiciones de salida para PNT RELLENA CAJA; HL debe contener la dirección de pantalla para la esquina superior izquierda de la caja, D debe contener el ancho de bytes de la caja, E debe contener las líneas de pantalla de la altura de la caja y C debe contener la tinta codificada.

Se inicia un bucle en 0DB7. HL se guarda en la pila, A = D, y se llama a 0EE8 para comprobar si las direcciones de la línea están en secuencia directa. Si no requieren corrección, 0EE8 regresa con el acarreo a cero, en cuyo caso una simple rutina LDIR se puede utilizar para activar los bytes de una línea. Este es el método rápido, pero no se puede utilizar en todos los casos. Si 0EE8 regresa con el acarreo a uno, la secuencia (HL) = C: PNT SIGUIENTE BYTE, se repetirá D veces.

En ambos casos se llama a PNT SIGUIENTE LINEA y la rutina ejecutará el bucle de 0DB7 E − 1 veces, encargándose así de todas las líneas de la caja.

PNT INVIERTE CARACTER *(SCR CHAR INVERT):* BC4A, 0DDF

A su entrada, B y C contienen diferentes tintas codificadas, H contiene un número de columna física, y L contiene un número de línea física. Los colores del carácter de la posición indicada son intercambiados.

C = B XOR C forma una máscara que indica los bits que son diferentes en los dos colores. El proceso (HL)=(HL) XOR C se aplica a todos los bytes que forman el carácter.

PNT DESPLAZA PANTALLA *(SCR HW ROLL):* BC4D, 0DFA

La pantalla puede desplazarse por simple modificación del offset si el área de la ventana abarca la totalidad de la pantalla. Este es el llamado desplazamiento *hardware*.

El contenido a la entrada de B determina la dirección del desplazamiento. B=0 dará un desplazamiento hacia abajo, mientras que cualquier otro valor de B indicará desplazamiento hacia arriba.

Primero, los 48 bytes no utilizados de la RAM de pantalla se ponen con el color de fondo. Estos formarán parte de la línea que se llevará al área visible de la pantalla. La rutina llama entonces a MC ESPERA BARRIDO antes de modificar el offset en ±&50 y antes de limpiar las 32 restantes posiciones de la nueva línea.

PNT DESPLAZA AREA *(SCR SW ROLL):* BC50, 0E3E

Si el cauce actual tiene definida una ventana más pequeña que el tamaño total de la ventana, se utilizará el desplazamiento *softwate*, permitiendo que las áreas exteriores a la ventana permanezcan inalteradas. Como con el desplazamiento *hardware*, el contenido de B a la entrada determinará la dirección del desplazamiento. B=0 dará un desplazamiento hacia abajo.

Existen rutinas separadas para las dos direcciones de desplazamiento, pero el principio es el mismo para ambas. la línea que se sale fuera de los márgenes del área se sobreescribe copiando en ella la siguiente línea, repitiéndose el proceso para las restantes líneas. Finalmente, la última línea se borra, por medio de PNT RELLENA BYTE.

Allí donde es posible, se emplea la instrucción LDIR para el proceso de copia, pero (al igual que en PNT RELLENA BYTE) esto no es posible si han de cruzarse líneas o bloques de los límites. Merece la pena resaltar que las manipulaciones de bloques de pantalla son mucho más rápidas cuando el offset es cero y no hay ventanas definidas.

PNT EXPANDE *(SCR UNPACK):* BC53, 0EF3

Una "matriz modelo" de carácter define la forma del carácter, pero sólo de manera directamente aplicable al modo 2. Para los otros modos, la matriz ha de ampliarse a 16 bytes para el modo 1 y a 32 bytes para el modo 2. Este proceso es conocido como "expansión" de la matriz.

A su entrada, HL debe apuntar al comienzo de un bloque de matriz de un carácter, el cual puede estar en el área 3800-3FFF de la ROM o en un área definida por el usuario en la RAM. DE debe apuntar a un área de la RAM lo suficientemente grande para contener la matriz expandida.

Para cada modo se presentan rutinas separadas. En el modo 2, la matriz se copia directamente, y sin modificación, en el área dispuesta para ello en la RAM.

Para el modo 1, la conversión es la siguiente:

Byte de la matriz	a	b	c	d	e	f	g	h
Primer byte expandido	a	b	c	d	a	b	c	d
Segundo byte expandido	e	f	g	h	e	f	g	h

La conversión está determinada por referencia a las máscaras de color en B1CF-B1D2, y la conversión es:

Byte de la matriz	a	b	c	d	e	f	g	h
Primer byte expandido	a	b	a	b	a	b	a	b
Segundo byte expandido	c	d	c	d	c	d	c	d
Tercer byte expandido	e	f	e	f	e	f	e	f
Cuarto byte expandido	g	h	g	h	g	h	g	h

El método general es desplazar a la izquierda bit a bit el byte de la matriz original y, si el bit transferido al acarreo es uno, entonces A=A OR (máscara de color). La máscara de color se cambia para cada desplazamiento y el proceso se repite para los ocho bytes de la matriz.

PNT COMPRIME *(SCR REPACK):* BC56, 0F49

El proceso inverso toma la matriz expandida de la RAM de pantalla, por lo que es necesario especificar la posición del carácter; H contiene la columna y L contiene la línea, ambas en coordenadas físicas. El color también debe ser tenido en cuenta, por lo que A debe contener la tinta codificada. DE apunta a un área de ocho bytes de la RAM, que es donde puede reconstruirse la matriz básica.

C=A y se llama a PNT POSICIONA COORDENADAS para convertir a HL en una dirección de pantalla; después la rutina se bifurcará, según el modo de pantalla actual.

Para el modo 2, la siguiente acción se repite para cada byte de la matriz:

A=(HL) XOR C
Se complementa A
(DE)=A
DE=DE+1
CALL PNT SIGUIENTE LINEA

Para los otros modos, A=(HL) OR C, y el resultado se compara con las máscaras de color del modo. Si A AND (máscara de color)=0, se desplaza un 1 en el byte de salida; en otro caso se desplaza un 0, desplazándose el byte a la izquierda, es decir, hacia el acarreo.

PNT MODO GRAFICOS *(SCR ACCESS):* BC59, 0C49

PNT ESCRIBE PIXEL puede trabajar en cuatro modos diferentes, y PNT MODO GRAFICOS determina cuál de los modos se va a utilizar. El contenido de A a la entrada de esta rutina determina el destino de un salto. Los cuatro modos, con B=tinta codificada y C=máscara del *pixel*, se dan a continuación con los valores de A que provocan la selección de cada uno de ellos por parte de PNT MODO GRAFICOS.

Modo forzado (A=0):

A=(HL)
A=A XOR B
A=A OR C
A=A XOR C
A=A XOR B
(HL)=A

Las implicaciones de esta acción se muestran mejor con una tabla de verdades·

(HL)	B	C	XOR B	OR C	XOR C	XOR B
0	0	0	0	0	0	0
0	0	1	0	1	0	0
0	1	0	1	1	1	0
0	1	1	1	1	0	1
1	0	0	1	1	1	1
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	1	0	1	0	1

Cuando C=1, la secuencia OR C, XOR C provocará un estado cero, mientras que, cuando C=0, el estado previo permanece inalterado. En el último caso, XOR B, XOR B no hace diferente a (HL), luego los bits solamente serán modificados con C=1, cuando el estado final depende únicamente del estado inicial d B.

Modo XOR (A=1):

A=B AND C
(HL)=A XOR (HL)

Los bits que estén a uno en B y C a la vez se invertirán en (HL).

Modo AND (A=2):

A=C complementado
A=A OR B
(HL)=A AND (HL)

Los bits de (HL) que corresponden a los bits a cero de B, emparejados con bits a uno en C, se pondrán a cero.

Modo OR (A=3):

A=B AND C
(HL)=(HL) OR A

Los bits que estén a uno en B y en C se pondrán a uno en (HL). Tal vez la mejor manera de comprender las implicaciones de estos modos sea experimentar directamente con ellos.

PNT PIXEL *(SCR PIXELS):* BC5C, 0C6B

Esta es idéntica a PNT ESCRIBE PIXEL en el modo forzado, excepto en que no es una indirección.

PNT HORIZONTAL *(SCR HORIZONTAL):* BC5F, 0FC4

Todas las líneas de gráficos se componen de segmentos horizontales o verticales. Esta rutina se encarga de dibujar los segmentos horizontales. A su entrada, A debe contener una tinta codificada; DE, la coordenada X

inicial; BC, la coordenada X final, y HL, la coordenada Y. Todas las coordenadas se refieren a desplazamientos absolutos desde la esquina inferior izquierda de la pantalla.

La rutina es larga, pero muy directa y comprensible. HL determina la línea del haz que se va a utilizar; DE y BC determinan los puntos extremos. Observa que éstos no son afectados por el último punto que se haya dibujado.

PNT VERTICAL *(SCR VERTICAL):* BC62, 102F

Esta es similar a PNT HORIZONTAL, pero es mucho más simple, ya que la posición de los puntos pertenece a una serie de líneas del haz de barrido de la pantalla. A su entrada, A debe contener una tinta codificada; DE, la coordenada X; HL, la coordenada Y inicial, y BC, la coordenada Y final.

Nos quedan todavía dos indirecciones que abordar:

PNT LEE PIXEL *(SCR READ):* BDE5, 0C82

A su entrada, HL contiene la dirección de pantalla de un byte, y una máscara en C identifica los *pixels* cubiertos por el byte. Estos pueden obtenerse desde PNT COORDENADA PUNTOS. A su salida, A contiene la tinta decodificada del *pixel*. La rutina consiste en A=(HL), seguido de la parte relevante de PNT DECODIFICA TINTA.

PNT ESCRIBE PIXEL *(SCR WRITE):* BDE8, 0C68

A su entrada, HL contiene la dirección de la pantalla de una posición de carácter; C contiene una máscara del *pixel*, y B contiene una tinta codificada. Con un salto a B1CC nos encontramos con otro salto inicializado por PNT MODO GRAFICOS. Desde aquí se trabaja con una de las cuatro rutinas de escritura ya examinadas.

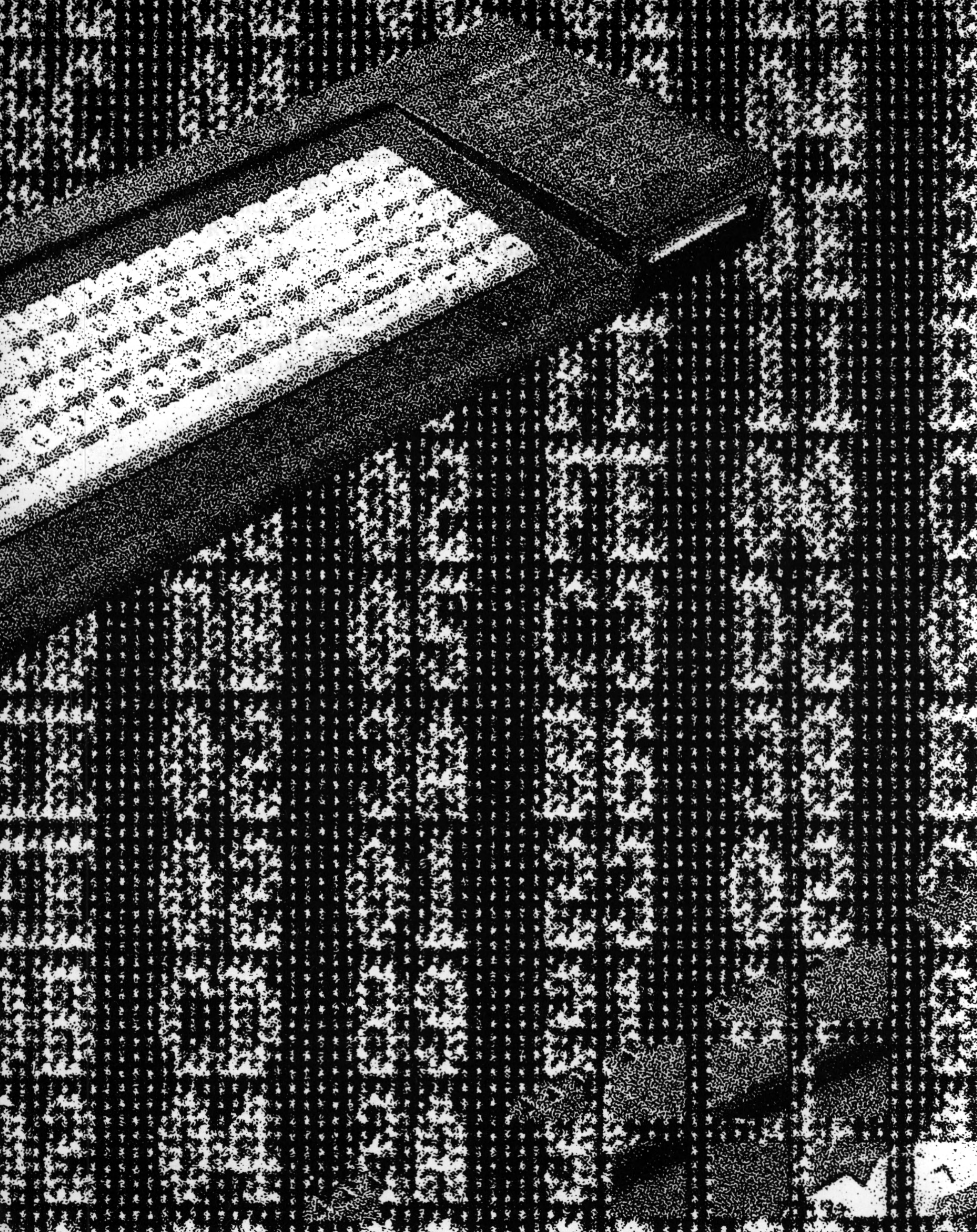
Comentarios

Las rutinas de control de pantalla no deberían ser juzgadas de forma aislada. Estas forman el caballo de batalla diseñado para servir a las funciones de más alto nivel, y precisamente éstas son las rutinas más utilizadas por las funciones de más alto nivel. Por ejemplo, PNT ESCRIBE

PIXEL fija únicamente un byte de pantalla, ya que puede ser necesario para fijar un solo *pixel*, mientras que la presentación de un carácter puede requerir la fijación de hasta 32 bytes.

Este es el trabajo de la VDU de texto, que también tiene que asegurarse de que los datos definen un carácter a presentar, y no un código de control a ejecutar.

El control de pantalla se encarga de las operaciones que implican el uso de la pantalla, principalmente como sirviente de la VDU de texto y de gráficos; pero hay veces en que se puede acceder a este control con cierta ventaja. En cualquier caso, se merece un estudio cuidadoso, ya que es necesaria su comprensión con base al examen de las rutinas de la VDU.



7
La unidad de video para texto

La unidad de video para texto (VDU de texto) ocupa la zona 1078-15A0 de la ROM y ofrece 36 entradas y 5 indirecciones. Su principal papel consiste en colocar los caracteres en la pantalla, encargándose también de los códigos de control.

TXT INICIALIZA *(TXT INITIALISE):* BB4E, 1078

Al comenzar se efectúa una llamada a TXT REINICIALIZA; entonces (B295)=0 indica que no hay ninguna matriz de usuario disponible. Se llama a 113D con HL=0001 para disponer:

(B28D)=3	Cursor inhibido y desactivado
(B28F)=H	Papel 0
(B290)=L	Pluma 1
(B293)=0	No escribe gráficos
(B291/2)=1391	Modo opaco
Ventana=Pantalla completa	
VDU ACTIVADA	

Esto determina el cauce actual y la rutina termina vía 10A3 para copiar los datos de este cauce en todos los cauces.

TXT REINICIALIZA *(TXT RESET):* BB51, 1088

Las cinco indirecciones han sido dispuestas en lugar de direcciones normales de entrada. La tabla de conexiones para los códigos de control se copia en el área B2C3-B322 de la RAM, a partir de 146B-14CA de la ROM.

Control de pantalla y cursor

TXT ACTIVA VDU *(TXT VDU ENABLE):* BB54, 1415

Al comenzar se llama a TXT ACTIVA CURSOR USUARIO, (B28E)=&FF, (B2B8)=0.

Tenemos que estudiar ahora un grupo de rutinas relativas a la posición del cursor. Resultan complicadas, ya que las coordenadas del usuario se expresan en términos "lógicos", contando desde 1, mientras que los datos almacenados están en términos "físicos", contando desde 0.

TXT FIJA COLUMNA *(TXT SET COLUMN):* BB6F, 115E

Al entrar, A debe contener el número de la columna lógica requerida. Añadiéndolo a la parte izquierda de la ventana y restando 1, se obtiene la columna absoluta. H=A, L=(B285), que es el número de la fila; entonces se llama a TXT CURSOR ELIMINADO. Después (B285/6)=HL, y la rutina sale vía TXT CURSOR ACTIVADO.

Observa que el cursor debe ser suprimido antes de cambiar las coordenadas.

TXT FIJA FILA *(TXT SET ROW):* BB72, 1174

Esta rutina es similar a la anterior, salvo que A debe especificar una fila requerida. L=A+(borde superior de la ventana)−1, H=(B286), que es la columna, seguido de la secuencia del cursor.

TXT EXAMINA CURSOR *(TXT GET CURSOR):* BB78, 1180

Literalmente, esta rutina devuelve la posición del cursor relativa a los límites de la ventana actual, pero la validez de la posición no se comprueba y, si ésta queda fuera de la ventana, podrá cambiarse antes de utilizarse.

HL=(B285/6)	coloca en H la columna absoluta y en L la fila absoluta.
H=H−(B298)+1	convierte la coordenada de la columna en coordenada del usuario.
L=L−(B288)+1	convierte la coordenada de la fila en coordenada del usuario.
A=(B28C)	recoge el valor de la cuenta de desplazamiento.

La cuenta de desplazamiento se decrementa en un desplazamiento hacia arriba y se incrementa si es hacia abajo. Sólo sirve para indicar si se ha producido un desplazamiento.

El control que permite activar el cursor depende de los bits 0 y 1 de (B28D). El bit 0 se controla por el usuario (INHIBE/DESINHIBE), mientras que el bit 1 se controla por el sistema (ON/OFF). Si cualquiera de los bits está a nivel bajo, el cursor no aparecerá.

TXT ACTIVA CURSOR USUARIO *(TXT CUR ENABLE):* BB7B, 1289

Se llama a TXT CURSOR ELIMINADO. Entonces (B28D)=(B28D) AND &FE, poniendo a cero el bit 0. La rutina termina vía TXT CURSOR ACTIVADO.

TXT INHIBE CURSOR USUARIO *(TXT CUR DISABLE):* BB7E, 129A

Es similar a la rutina anterior, salvo que (B28D) =(B28D)OR 1, colocando a uno el bit 0.

TXT ACTIVA CURSOR STMA *(TXT CUR ON):* BB81, 1279

Igual que TXT ACTIVA CURSOR USUARIO, salvo que el bit 1 de (B28D) se pone a 0.

TXT INHIBE CURSOR STMA *(TXT CUR OFF)*: BB84, 1281

Igual que TXT INHIBE CURSOR USUARIO, salvo que se pone a uno el bit 1 de (B28D).

Las dos siguientes rutinas preservan AF.

Se nos presenta ahora un problema: dos entradas del bloque de saltos que llaman al mismo punto de entrada, aunque su acción es equivalente y opuesta. Si el cursor está ausente, se le hará aparecer, y si está presente, desaparecer, mediante inversión del carácter que esté en la posición del cursor.

TXT PONE CURSOR *(TXT PLACE CURSOR)*: BB8A, 1268

TXT QUITA CURSOR *(TXT REMOVE CURSOR)*: BB8D, 1268

BC, DE y HL son preservados. Una subrutina situada en 11AB carga a HL con el contenido de (B285/6), que son la fila y columna de la posición del cursor; comprueba la validez de la posición en relación con la ventana actual, mediante la subrutina situada en 11DA (véase TXT POSICION VALIDA), y vuelve a cargar (B285/6) con el resultado de esta operación que está contenido en HL, y que puede haber sido modificado para adaptar la posición a la ventana. Si el acarreo está puesto a uno, la subrutina de 11AB retorna. De otro modo, se requiere un desplazamiento. Si B≠0, entonces A=1, mientras que si B=&FF, entonces A=&FF. A se añade a la cuenta de desplazamiento; B indica la dirección requerida del desplazamiento.

A continuación se llama a TXT EXAMINA VENTANA. Si retorna cuando el acarreo está puesto a 1, se llama a PNT DESPLAZA AREA, y si no, se llama a PNT DESPLAZA PANTALLA. En ambos casos A=(B290), que es el papel.

De vuelta a la rutina principal, B=pluma, C=papel, y se llama a PNT INVIERTE CARACTER para invertir el carácter que está en la posición del cursor.

El antiguo cursor debe haber sido trasladado previamente, utilizando la misma rutina con la antigua posición del cursor.

En conexión con esta rutina de doble finalidad, tenemos un par de indirecciones:

TXT CURSOR ACTIVADO *(TXT DRAW CURSOR):* BDCD, 1263

TXT CURSOR ELIMINADO *(TXT UNDRAW CURSOR):* BDD0, 1263

Si (B28D) es distinto de 0, la rutina regresa y no realiza ninguna acción. En otro caso, seguirá actuando TXT PONE CURSOR.

TXT POSICION VALIDA *(TXT VALIDATE):* BB87, 11CE

Esta rutina comprueba si la posición del cursor queda comprendida dentro de la ventana actual y reajusta esta posición si resulta fuera de la pantalla. Al entrar, H contiene la columna y L la fila, en coordenadas lógicas tomadas desde 1. Las coordenadas se convierten en coordenadas absolutas añadiendo el borde izquierdo de la ventana menos uno y el borde derecho de la ventana menos uno. Una subrutina situada en 11DA comprueba y modifica del modo siguiente:

- Si H es superior al borde derecho de la ventana, entonces H es igual al borde izquierdo de la ventana, L=L+1.
- Si H es inferior al borde izquierdo de la ventana, entonces H es igual al borde derecho de la ventana, L=L−1.
- Si L es inferior al borde superior de la ventana, entonces L es igual al borde superior de la ventana, B=0, y la rutina regresa con el acarreo a cero y B=0 para ordenar un desplazamiento hacia abajo. De otro modo, si L es superior al borde inferior de la ventana, volverá con el acarreo puesto a uno. Además de esto, L será igual al borde inferior de la ventana y la rutina volverá con el acarreo puesto a cero y B=&FF para obtener un desplazamiento hacia arriba.

HL es reconvertido en coordenadas lógicas. Si el acarreo está puesto a cero, se ejecuta el desplazamiento.

TXT COLOR PLUMA *(TXT SET PEN):* BB90, 12A9

TXT COLOR PAPEL *(TXT SET PAPER):* BB96, 12AE

Además de disponer HL = B28F para la pluma, y HL = B290 para el papel, estas entradas utilizan una rutina común. Al comenzar, A contiene la tinta elegida. Se llama a TXT CURSOR ELIMINADO y después a PNT CODIFICA TINTA. (HL) = A, y la rutina sale vía TXT CURSOR ACTIVADO.

El cursor debe eliminarse, ya que el cambio de color trastornaría en otro caso el proceso de inversión.

TXT EXAMINA PLUMA *(TXT GET PEN):* BB93, 12BD

TXT EXAMINA PAPEL *(TXT GET PAPER):* BB99, 12C3

Para la pluma, A = (B28F); para el papel, A = (B290). A continuación le seguirá PNT CODIFICA TINTA.

TXT INVIERTE COLORES *(TXT INVERSE):* BB9C, 12C9

La pluma (B289) y el papel (B290) se intercambian.

TXT FIJA MODO *(TXT SET BACK):* BB9F, 137A

Esta rutina determina si el color de fondo debe ser escrito (opaco) o se mantiene inalterado (transparente). La acción se determina colocando una dirección de enlace:

Si A=0, (B291/2)=1391, un enlace para opaco.
Si A=1, (B291/2)=139F, un enlace para transparente.

La rutina elegida se llama mediante TXT ESCRIBE CARACTER, pero es conveniente describir aquí la acción. C contiene el byte matriz, y DE contiene la dirección de pantalla.

Opaco

HL=(B28F/90), pluma y papel.
B=H AND (C complementado), la máscara de papel.
A=C AND L, máscara de la pluma.
C=&FF ordena que todos los puntos de pantalla se pongan a uno.
B=A OR B, máscara combinada.
EX DE, HL pone la dirección de la pantalla en HL.
Le sigue PNT PIXEL.

Transparente

B=(B28F), pluma.
EX DE, HL.
Le sigue PNT PIXEL.

Observa que sólo está disponible el modo forzado.

TXT EXAMINA MODO *(TXT GET BACK):* BBA2, 1387

Si (B291/2)+EC6F=0, el enlace dispuesto es para el modo opaco, y la rutina retorna con A=0. Si el enlace se coloca para transparente, A≠0.

Ventanas

TXT ACTIVA VENTANA *(TXT WIN ENABLE):* BB66, 120C

A la entrada, D y H deben contener las columnas físicas para las posiciones de los lados de la ventana, considerando a la mayor como definidora del lado derecho. De modo similar, E y L contienen las filas superior e inferior, siendo la mayor la que define la fila inferior. Se

comprueba la validez de las coordenadas, ajustándose si resultan fuera del área de la pantalla. Los valores resultantes se colocan en (B288/B) (véase rutina siguiente).

Si la ventana cubre el total de la pantalla, (B287)=0, siendo éste el *flag* que selecciona el desplazamiento *hardware* o *software*. La posición del cursor se coloca en la esquina superior izquierda de la pantalla.

TXT EXAMINA VENTANA *(TXT GET WINDOW)*: BB69, 1256

L=(B288)	Borde superior
H=(B289)	Borde izquierdo
E=(B28A)	Borde inferior
D=(B28B)	Borde derecho

Si (B287)=0, la rutina regresa con el acarreo puesto a cero, permitiendo el desplazamiento *hardware*. Si el acarreo está puesto a 1, se requerirá el desplazamiento *software*.

TXT LIMPIA VENTANA *(TXT CLEAR WINDOW)*: BB6C, 1540

El cursor se "elimina", (B285/6)=(B288/9), colocándolo en la posición inicial, en la esquina superior izquierda de la ventana; HL y DE se disponen como para TXT EXAMINA VENTANA, vista hace un momento, y se llama a PNT RELLENA CAJA. Si está activado, el cursor es restaurado.

Los múltiples cauces, cada uno de los cuales puede tener su propia serie de parámetros, complican algunas de las rutinas que hemos visto. Las manipulaciones del cursor también revisten mayor complejidad de la que cabría esperar.

Por otro lado, la flexibilidad del sistema de pantalla justifica esta complejidad, por lo que merece la pena un estudio profundo del sistema.

Cauces

Se han utilizado mucho los datos de la pantalla actual en el área (B285-B293). En cualquier momento, los datos para otro cauce se pueden copiar dentro de esta área, desde las copias contenidas en la zona (B20D-B284). Los datos anteriores se protegen en su área de copia.

TXT SELECCIONA CAUCE *(TXT STREAM SELECT):* BBB4, 10E8

A la entrada, A contiene el número del cauce requerido. A=A AND 7 asegura que se está utilizando un número entre 0 y 7. Si A=(B20C), que es el número del cauce actual, la rutina regresa sin ejecutar acción alguna.

En otro caso, BC y DE son guardados en la pila, C=(B20C), y (B20C)=A, cambiando el número del cauce actual. B=A, y A=C. DE=B20D+15*A coloca la dirección del área de copia para el cauce actual y HL=B285; BC=000F son preparados para proteger los parámetros actuales en el área de copia mediante LDIR.

A es cargado entonces con el número del nuevo cauce, y el proceso de copia se repite, aunque con un intercambio de DE y HL para invertir la dirección de transferencia. A se carga con el número del cauce anterior.

TXT CANJEA CAUCES *(TXT SWAP STREAMS):* BBB7, 1107

Esta rutina comporta algunos trucos interesantes. A la entrada, B y C contienen los números de dos cauces que son los datos que se van a intercambiar.

A=(B20C) anota el número del cauce actual, y AF se guarda en la pila.

Se llama a TXT SELECCIONA CAUCE con A=C para almacenar los datos del cauce actual y traer el dato C del cauce al área actual. Entonces (B20C)=B, y DE es cargado con B20D+15*B. DE se guarda en la pila, A=C, y DE=B20D+15*C. La dirección guardada en la pila por DE se recupera en HL y la rutina de copia transfiere el dato B del cauce al área C del mismo. AF es recuperado de la pila y TXT SELECCIONA CAUCE es llamada para copiar el dato C del anterior cauce (en el área común) en el área B del cauce (porque (B20C)=B). El cauce definido en A, que era actual antes de que este proceso se iniciara, se trae al área actual.

Datos de las matrices

Los modelos de matriz del carácter estándar están contenidos en la ROM, en la zona 3800-3FFF, pero los modelos especiales pueden definirse en la RAM por el usuario. Cuando dichos modelos de RAM se hayan definido, el *flag* de matriz situado en (B295) será distinto de 0, y (B294)

contendrá el código para el primer carácter de la tabla del usuario. La dirección de comienzo de la tabla de la RAM está contenida en (B296/7).

Cuando se busca un modelo, podrá estar en la ROM o en la RAM, y la elección se realiza mediante:

TXT EXAMINA TABLA MATRICES *(TXT GET M TABLE):* BBAE, 132A

HL=(B294/5), poniendo el *flag* de la matriz en H y el código para el primer carácter definible por el usuario en L. Si H≠0, el acarreo se pone a uno. Entonces, HL=(B296/7), y éste será el comienzo de la tabla definible por el usuario, mientras no se establezca otro. Si el acarreo está puesto a cero, HL es ignorado, pero en cualquier caso A=L.

TXT EXAMINA MATRIZ *(TXT GET MATRIX)*: BBA5, 12D3

A la entrada, A contiene un código de carácter.

12D3 DE se guarda en la pila, E=A, y se llama a TXT EXAMINA TABLA MATRICES. Si regresa con el acarreo puesto a cero, la rutina salta a 12E3. No hay tabla de usuario. Si, en otro caso, E es menor que el valor de A dispuesto por TXT EXAMINA TABLA MATRICES, el código estará por debajo del intervalo definido para la tabla del usuario y la rutina también saltará a 12E3. De otro modo, se seguirá 12E6 con E=E−A.

12E3 HL=3800, que es la base de la tabla de la ROM.

12E6 HL=HL+8*E, lo que constituye la dirección de la matriz requerida. Se recupera DE y la rutina regresa.

Para crear una tabla matriz del usuario, el área de la RAM debe ser previamente definida y los datos relevantes deben ser inicializados.

TXT FIJA TABLA DE MATRICES *(TXT SET M TABLE):* BBAB, 12FD

A la entrada, E contiene el código para el primer carácter de la tabla y D es igual a 0. Sin embargo, si D≠0, la tabla existente se cancela. HL contiene la dirección de comienzo de la tabla.

12FD HL se guarda en la pila, protegiendo la dirección de comienzo de la tabla de la RAM. Si D≠0, la rutina saltará a 131D con D=0, lo que provoca que el *flag* de la matriz se ponga a cero. En otro caso, D=&FF y DE se guarda en la pila. C=E, mientras que DE y HL se intercambian.

1308 A=C, y se llama a TXT EXAMINA MATRIZ. Si, como es de esperar, el *flag* de la matriz está puesto a cero, la dirección en la ROM de la matriz que corresponde al código de carácter contenido a la entrada en E se cargará en HL. Si HL=DE, la rutina saltará a 131C. También:

1314 BC se guarda en la pila y se copian 8 bytes de (HL) a (DE) mediante LDIR; BC se recupera y C se incrementa para apuntar al siguiente carácter. Si C≠0, la rutina enlaza con 1308, iniciándose un bucle.

131C Se recupera DE.

131D Se llama a TXT EXAMINA TABLA MATRICES y entonces DE se copia en (B294/5), cargándose el *flag* de la matriz desde D y el primer código de carácter desde E. La dirección de comienzo de la nueva tabla se recupera entonces y se carga en (B296/7).

Esto prepara la entrada de la tabla del usuario, que consiste en una copia de la parte apropiada de la tabla de la ROM. Ahora deben colocarse las matrices del usuario:

TXT IMPONE MATRIZ *(TXT SET MATRIX):* BBA8, 12F1

La matriz se coloca copiando de una fuente determinada, que podría ser otra matriz existente. A la entrada, A contiene el código para la matriz y HL apunta a la fuente de copia. Si A regresa con el acarreo puesto a cero, significará que ha fracasado; si está puesto a uno, se entenderá que llegó a buen fin. El fracaso indica que el carácter no se encuentra dentro de la tabla del usuario definida, o que no se ha definido tabla alguna.

DE=HL, y se llama a TXT EXAMINA MATRIZ. Si al regreso C=0, la rutina termina. Si no, DE y HL se intercambian, y la LDIR copia la matriz en su sitio.

Salida de textos

El sistema de salida de textos se complica debido a las interconexiones de las llamadas apropiadas. TXT SALIDA utiliza TXT CODIGO CARACTER (una indirección). TXT CODIGO CARACTER maneja los códi-

gos de control, pero envía códigos de carácter a TXT ESCR CARACTER o al sistema de gráficos. TXT ESCR CARACTER llama a TXT ESCRIBE CARACTER, otra indirección.

Pero esta complicación también tiene sus ventajas. Por ejemplo, GRA ESCRIBE CARACTER no responde a los caracteres de control, sino que los saca en pantalla, lo que puede ser enojoso. Alterando la indirección TXT CODIGO CARACTER, podemos interceptar los códigos de control y tratarlos con mayor eficacia.

Las rutinas estarán descritas en orden inverso, desde el final de la cadena, marcha atrás, hasta el principio, ya que de este modo se ve más clara la acción.

TXT ESCR CARACTER *(TXT WRITE CHAR):* BDD3, 134A

A la entrada, A contiene un código ASCII, H contiene una columna física, y L contiene una fila física (contando a partir de 0).

134A HL es guardado en la pila, y se llama a TXT EXAMINA MATRIZ para obtener la dirección de la matriz requerida. DE=B298, que es el comienzo del área tomada como modelo. DE se guarda en la pila y se llama a PNT EXPANDE para extender la matriz en el área elegida. DE y HL son recuperados y se llama a PNT POSICIONA COORDENADAS para colocar una dirección de pantalla. C=8.

135C BC y HL se guardan en la pila. Esto es un punto de un bucle externo.

135E BC y DE se guardan en la pila. Esto es un punto de un bucle interno. DE y HL se intercambian, C=(HL), y llamando a 1376 se accede a los modos opaco o transparente (véase TXT FIJA MODO). Se llama a PNT SIGUIENTE BYTE, y DE se recupera y se incrementa. BC es recuperado, y seguirá un DJNZ hacia 135E. Este suele activar una línea. (Advierte que B se carga con el número de bytes contenidos en el ancho de un carácter, mediante PNT POSICIONA COORDENADAS.) Cuando el bucle DJNZ finaliza, HL se recupera, se llama a PNT SIGUIENTE LINEA, BC se recupera, C se decrementa; si C≠0, la rutina enlaza con 135C y, si C=0, la rutina regresará, habiendo ya dispuesto las ocho líneas.

TXT ESCRIBE CARACTER *(TXT WR CHAR):* BB5D, 1334

A la entrada, A contiene un código ASCII. B=A. Si (B28E)=0, la rutina regresa inhibiendo la imagen. En caso contrario, BC se guarda en la pila, y se llama a la rutina en 11A8. Esto traslada el cursor, comprueba la validez y, si es necesario, corrige los valores de columna y de línea para que resulten dentro de la ventana actual. (B285)=H+1, colocando la posición de la siguiente columna, y AF se recupera, rescatando de B el código guardado en la pila. TXT ESCR CARACTER es llamada, y luego TXT CURSOR ACTIVADO.

TXT SALIDA *(TXT OUTPUT):* BB5A, 1400

Se llama a TXT CODIGO CARACTER, habiendo guardado en la pila los registros BC, DE, HL y AF.

TXT CODIGO CARACTER *(TXT OUT ACTION):* BDD9, 140C

A la entrada, A contiene un valor que podrá ser tanto un código de carácter como un código de control, e incluso un parámetro que siga a un código de control. La rutina debe decidir de cuál de estas posibilidades se trata.

Si el parámetro contenido en (B2B8)=0, el valor no será un parámetro, sino un código ASCII. Si es igual o mayor que &20, se tratará de un código de carácter; si es menor que &20, será un código de control.

Si el *flag* de escritura de gráficos en (B293)≠0, la acción pasa a GRA ESCRIBE CARACTER, tanto si el valor representa un código de control como si no. Esto explica por qué la escritura de gráficos insiste en mostrar los símbolos de los códigos de control de forma tan desconcertante.

La siguiente descripción debería leerse conjuntamente con la tabla de códigos de control proporcionada más adelante. Esto detalla e interpreta los datos contenidos en la RAM desde B2C3 en adelante.

La primera acción es copiar A en C y comprobar el *flag* de escritura de gráficos, saltando a GRA ESCRIBE CARACTER con A=C si el *flag* no está puesto a cero.

El valor del parámetro en B2B8 es comprobado y si (B2B8) toma un valor mayor que 9, la rutina sale con (B2B8)=0. Algo ha debido extraviarse, puesto que ningún código de control requiere más de nueve paráme-

tros. Si (B2B8)=0 y A contiene un valor mayor que &1F, la rutina salta a TXT ESCRIBE CARACTER.

En otro caso, B=(B2B8)+1 y se forma una dirección como B2B8+B. El valor contenido en A se almacena en esta dirección. Observa que si el valor es menor que &20, (B2B8) debe ser igual a cero y el código de control se almacena en (B2B9); pero si (B2B8)≠0, el valor es un parámetro y se almacenará entre (B2B9) y (B2C1).

El contenido de (B2B9), que es el código de control, se utiliza para formar una dirección B2C3+3*(B2B9). Esto elige uno de los grupos de bytes contenidos en la tabla de códigos de control. El primer byte del grupo especifica el número de bytes del parámetro requerido. Si ese número no se ha encontrado, la rutina termina.

En caso contrario, se llama a la rutina indicada por el segundo y tercer bytes del grupo. Entonces (B2B8)=0, y la rutina en su conjunto regresará.

Tabla de carácter de control

Código	*Parámetros*	*Enlace*	*Función*
&00	0	14E2	Regreso inmediato. Sin acción
&01	1	1334	TXT ESCRIBE CARACTER
&02	0	139A	TXT INHIBE CURSOR USUARIO
&03	0	1289	TXT ACTIVA CURSOR USUARIO
&04	1	0ACA	PNT FIJA MODO
&05	1	1945	GRA ESCRIBE CARACTER
&06	0	1451	TXT ACTIVA VDU
&07	0	14D8	COLA SONIDO CON HL=14CF (pitido)
&08	0	150A	CURSOR IZQUIERDA
&09	0	150F	CURSOR DERECHA
&0A	0	1514	CURSOR ABAJO
&0B	0	1519	CURSOR ARRIBA
&0C	0	1540	TXT LIMPIA VENTANA
&0D	0	1530	Columna=Borde izquierdo ventana
&0E	1	12AE	TXT COLOR PAPEL
&0F	1	12A9	TXT COLOR PLUMA
&10	0	154F	BORRAR
&11	0	158E	LIMPIA VENTANA A LA IZQUIERDA DEL CURSOR
&12	0	1584	LIMPIA VENTANA A LA DERECHA DEL CURSOR
&13	0	156D	LIMPIA VENTANA ENCIMA DEL CURSOR
&14	0	1556	LIMPIA VENTANA DEBAJO DEL CURSOR

&15	0	144B	TXT DESACTIVA VDU
&16	1	14E3	TXT FIJA MODO
&17	1	0C49	PNT MODO GRAFICOS
&18	0	12C9	TXT INVIERTE COLORES
&19	9	1504	TXT IMPONE MATRIZ
&1A	4	14F8	TXT ACTIVA VENTANA
&1B	0	14E2	Regreso inmediato, sin acción
&1C	3	14E8	PNT IMPONE TINTA
&1D	2	14F1	PNT IMPONE BORDE
&1E	0	152A	CURSOR A ESQUINA SUPERIOR IZQUIERDA
&1F	2	1538	POSICIONA CURSOR

En muchos casos las rutinas se describen en otros sitios, pero las rutinas pueden no entrar directamente allí donde se precisan parámetros, puesto que éstos se eligen previamente en los registros apropiados. Para los movimientos del cursor, éste puede eliminarse, modificar su columna y/o fila, y se puede restaurar. Para borrar y aclarar grandes espacios se utiliza PNT RELLENA CAJA. Observa que la tabla está en la RAM, por lo que pueden realizarse fácilmente alteraciones, atribuyendo nuevos significados a los códigos de control.

Estas rutinas para implementar los códigos de control son bastante sencillas en general, por lo que no estimo necesario examinarlas aquí con detalle.

Una rutina similar sería:

TXT EXAMINA TABLA CONTROL *(TXT GET CONTROLS):* BBB1, 14CB

HL se coloca en B2C3, que es el comienzo de la tabla de códigos de control.

Otras rutinas de texto

Para el sistema editor, se necesita poder leer un carácter en la pantalla y obtener el correspondiente código ASCII. Hay dos rutinas que resultan adecuadas para ello:

TXT LEE CARACTER *(TXT READ CHAR):* BB60, 13AB

HL, DE y BC se guardan en la pila, y se llama a TXT CURSOR ELIMINADO, para que el carácter que esté en el cursor no se invierta, lo que complicaría las cosas. TXT LEE CAR es llamada con HL=(B285/6), fila/columna. AF se guarda en la pila y se llama a TXT CURSOR ACTIVADO. Los registros AF, BC, DE y HL son recuperados.

TXT LEE CAR *(TXT UNWRITE):* BDD6, 13C0

Esta es una indirección. A la entrada, H debe contener una columna física y L una fila física, contando a partir de 0. La matriz expandida se transfiere desde la posición de pantalla así definida al área en cuestión que comienza en B298, utilizando PNT COMPRIME introducida con A=(B28F), que es la pluma. Se llama a una rutina de cotejo, y si regresa con el acarreo puesto a uno, la rutina sale en su totalidad, con el registro A conteniendo el código ASCII requerido.

La rutina de cotejo compara la matriz comprimida con todas las matrices fuente, de una en una. Si no se encuentra pareja, la rutina de cotejo regresa con el acarreo a cero y el *flag* Z a uno. En este caso se vuelve a llamar a PNT COMPRIME, pero esta vez con A=(B290), que es el papel. La matriz resultante se invierte y la rutina de cotejo se reintroduce. Esto permite detectar caracteres en una tinta inesperada.

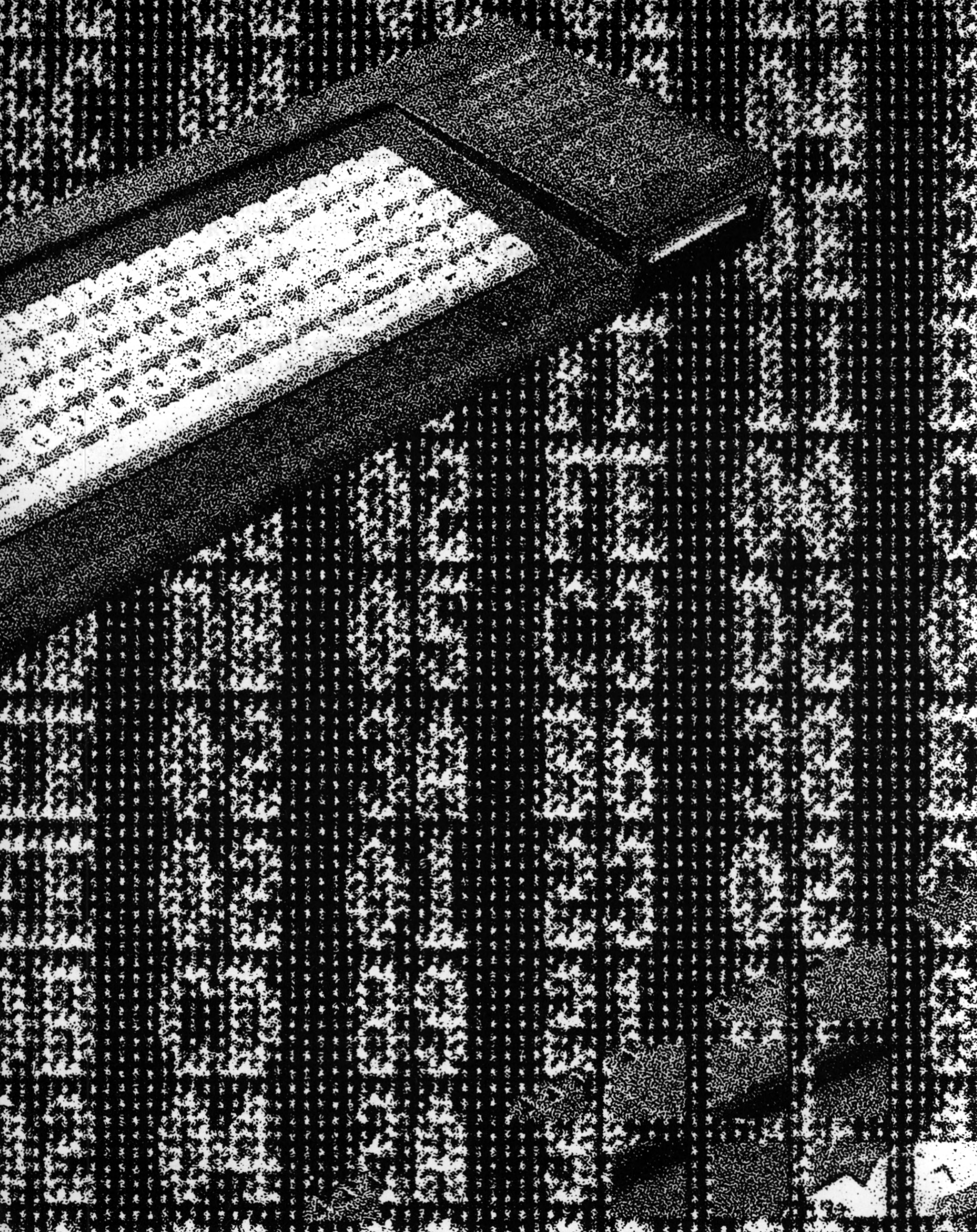
Queda un último punto de entrada.

TXT ADMISION DE GRAFICOS *(TXT SET GRAPHIC):* BB63, 137A

(B293)=A. Si A=0, se inhibe la escritura de gráficos, y se desinhibe en caso contrario.

Comentarios

Las rutinas de texto son quizá algo más complicadas de lo que cabría esperar, aunque gran parte de la acción resulte sencilla. El empleo de la inversión para crear el cursor no es del todo un éxito, ya que el carácter resultante es a veces difícil de leer, pero se ha proporcionado suficiente información como para permitir y estimular la experimentación de alternativas, como, por ejemplo, un simple subrayado. Las rutinas necesarias pueden obtenerse vía TXT CURSOR ACTIVADO y TXT CURSOR ELIMINADO.



8
La unidad de video para gráficos

La unidad de video para gráficos (VDU de gráficos), que ocupa el área 15B0-19DC de la ROM, proporciona 23 puntos de entrada y tres indirecciones.

Dado que los distintos modos de expresar las coordenadas pueden dar lugar a confusiones, creo conveniente ofrecer alguna explicación preliminar.

La posición actual del cursor de gráficos se almacena en (B32C/D) para X y en (B32E/F) para Y. los valores se dan en "coordenadas de usuario".

El origen se sitúa inicialmente en 0,0 —la esquina inferior izquierda de la pantalla—, pero puede alterarse a voluntad. Las coordenadas del origen están contenidas en (B328/9) para X y en (B32A/B) para Y.

Al entrar, un tipo absoluto de rutina coloca la posición actual desde el contenido de DE (X) y de HL (Y). Una clase relativa de rutina añade las coordenadas de la posición actual a DE y HL, introduciéndose entonces el tipo absoluto, la suma se realiza por una rutina en 1657.

Para fines internos se deberán modificar las coordenadas, dividiendo la coordenada X, que estará en forma de número entero, por el número de bits que representen un punto de pantalla en el modo actual. Si el número original es negativo, se aplicará un incremento. La coordenada absoluta Y se divide por dos, puesto que debe apuntar a una de las 200 líneas de pantalla.

Antes de llevar a cabo estas operaciones, las coordenadas del usuario se añaden a las coordenadas del origen. La subrutina 161D realiza esta conversión, entrando en 161A, previa llamada a GRA POSICION CURSOR.

Ahora ya podemos empezar a trabajar con las rutinas de inicialización.

GRA INICIALIZA *(GRA INITIALISE):* BBBA, 15B0

Se llama a GRA RE-INICIALIZA y, entonces, papel=0, pluma=1, origen=0,0 y la ventana de gráficos se define como la pantalla en su totalidad.

GRA RE-INICIALIZA *(GRA RESET):* BBBD, 15DF

Las indirecciones para GRA TEXT PUNTO, GRA LINEA y GRA PUNTO se fijan con las direcciones prescritas para omisiones.

Actualización

GRA FIJA ORIGEN *(GRA SET ORIGIN):* BBC9, 1604

A la entrada, DE debe contener la coordenada X requerida y HL la coordenada Y. Son valores absolutos. Las coordenadas del origen contenidas en (B328/B) se cargan desde DE y HL, que entonces se ponen a cero. Seguirá GRA MOV ABSOLUTO, que pone a cero las coordenadas de la posición actual en (B32C/F), moviendo efectivamente el cursor hasta el nuevo origen.

GRA ANCHURA VENTANA *(GRA WIN WIDTH):* BBCF, 1734

A la entrada, DE y HL deben contener las coordenadas estándar (absolutas) de los bordes izquierdo y derecho de la pantalla de gráficos. Las de mayor valor definirán el borde derecho. Los valores están limitados, de forma que queden dentro de los extremos de la pantalla y se transforman en coordenadas internas antes de ser almacenados en (B330/1), para el borde izquierdo, y (B332/3) para el borde derecho.

GRA ALTURA VENTANA *(GRA WINDOW HEIGHT):* BBD2 1779

Esta rutina se asemeja a la anterior, con la salvedad de que DE y HL contienen las coordenadas del borde superior e inferior de la ventana. Limitadas y transformadas, las coordenadas se almacenan en (B334/5) para el borde superior, y (B336/7) para el borde inferior.

GRA LIMPIA VENTANA *(GRA CLEAR WINDOW):* BBDB, 17C5

Se llama a GRA OBTIENE ANCHURA (véase después) para obtener las coordenadas de los bordes izquierdo y derecho de la ventana, a partir de las cuales se determina la anchura de la misma. El número de líneas de pantalla que forman la altura de la ventana se calcula de modo análogo. Se llama a PNT COORDENADA PUNTOS, A se carga desde (B339), que es el papel, y se llama a PNT RELLENA BYTE para realizar el borrado de la ventana. Le sigue la localización del cursor en su posición básica.

GRA COLOR PLUMA *(GRA SET PEN):* BBDE, 17F6

Se llama a PNT CODIFICA TINTA para codificar la tinta cargada en A, al entrar, y el resultado se almacena en (B338).

GRA COLOR PAPEL *(GRA SET PAPER):* BBE4, 17FD

Igual que la rutina anterior, pero el resultado se almacena en (B339).

Comprobando valores

GRA POSICION CURSOR *(GRA ASK CURSOR):* BBC6, 15FC

DE se carga desde (B32C/D) para proporcionar la coordenada actual X, y HL se carga desde (B32E/F) para proporcionar la coordenada Y.

GRA OBTIENE ORIGEN *(GRA GET ORIGIN):* BBCC, 1612

DE se carga desde (B328/9), que es la coordenada X del origen, y HL se carga desde (B32A/B), que es la coordenada Y.

GRA OBTIENE ANCHURA *(GRA GET W WIDTH):* BBD5, 17A6

DE se carga desde (B330/1), borde izquierdo, y HL desde (B332/3), borde derecho. Los valores se adaptan al modo correspondiente, dándose coordenadas absolutas.

GRA EXAMINA PLUMA *(GRA GET PEN):* BBE1, 1804

A=(B338), y a continuación se llama a PNT DECODIFICA TINTA.

GRA EXAMINA PAPEL *(GRA GET PAPER):* BBE7, 180A

A=(B339), y a continuación se llama a PNT DECODIFICA TINTA.

Funciones principales

Hasta aquí las rutinas han sido en cierto modo triviales, pero en ningún caso esenciales. Vamos a ver ahora las rutinas que realizan funciones fundamentales.

GRA MOV ABSOLUTO *(GRA MOVE ABSOLUTE):* BBC0, 15F4

GRA MOV RELATIVO *(GRA MOVE RELATIVE):* BBC3, 15F1

Para la versión relativa, se llama a la subrutina 1657 y luego se introduce la versión absoluta. (B32C/D)=DE establece la coordenada X y (B32E/F) dispone la coordenada Y. Los nuevos valores surtirán efecto cuando se llame a la siguiente función principal.

GRA PUNTO ABSOLUTO *(GRA PLOT ABSOLUTE):* BBEA, 1813

GRA PUNTO RELATIVO *(GRA PLOT RELATIVE):* BBED, 1810

GRA PUNTO *(GRA PLOT):* BDDC, 1816

Para la versión relativa, se llama a la subrutina 1657, y después le sigue la versión absoluta. Esta última llama inmediatamente a GRA PUNTO, que es una indirección.

Se llama a 16FC para adaptar al modo las coordenadas contenidas en DE y HL, utilizando 161D, y entonces realiza una función de comprobación, comparando la posición solicitada con los límites de la ventana. Si la posición no es válida, la rutina termina. En otro caso, se llama a PNT COORDENADA PUNTOS, B=(B338) y, a continuación, se llama a PNT ESCRIBE PIXEL.

Observa que este último paso supone la colocación de los bits relacionados con un punto en particular, pero también se puede utilizar la función normal de impresión.

GRA TEST PUNTO REL *(GRA TEST RELATIVE):* BBF3, 1824

GRA TEXT PUNTO *(GRA TEXT):* BDDF, 182A

Estas rutinas están relacionadas de modo similar al grupo anterior, soportando la indirección GRA TEXT PUNTO la acción principal. Se llama a 16FC para comprobar la validez de la posición requerida y, si regresa con el acarreo a cero, indicando una posición inválida, la rutina saldrá vía GRA EXAMINA PAPEL.

En otro caso la rutina saldrá vía PNT COORDENADA PUNTOS y PNT LEE PIXEL.

GRA LINEA ABSOLUTA *(GRA LINE ABSOLUTE):* BBF6, 1839

GRA LINEA RELATIVA *(GRA LINE RELATIVE):* BBF9, 1836

GRA LINEA *(GRA LINE):* BDE2, 183C

Aquí es donde el sistema de gráficos debe ganarse la vida trabajando. Las rutinas son demasiado complejas como para examinarlas con detalle, pero el núcleo de la acción es el siguiente:

A la entrada, DE contiene la coordenada X del punto final, mientras que HL contiene la coordenada Y. El punto inicial será la posición que ocupe en ese momento el cursor. El primer paso será calcular las distancias al punto (X, Y), y la proporción en la que las dos coordenadas deberán cambiar según se vaya dibujando la línea. Supongamos que la distancia a X, "AX", es la mayor. Entonces AX se divide por "AY", que es la medida de Y, y el resultado indicará cuántos incrementos de X deben tomarse por cada incremento de Y. El cálculo se hace sobre una base entera, utilizando una rutina discutida en el capítulo "Soporte del BASIC", pero se repite después de cada incremento de Y para minimizar los posibles errores resultantes. Supongamos que el resultado de la operación

es 5. Entonces se requerirá una pequeña línea horizontal de 5 unidades de largo, para empezar. Esto lo dibujo PNT HORIZONTAL.

AX se reduce entonces en 5; AY en 1. El proceso se repite para tomar un tramo nuevo de la línea, hasta que AY=0, AX=0.

GRA ESCRIBE CARACTER *(GRA WR CHAR):* BBFC, 1945

A la entrada, A contiene un código ASCII, aunque no necesariamente un código normal de carácter. IX se protege en la pila y se llama a TXT EXAMINA MATRIZ. DE=B33A, que es el principio del área de la matriz de copia, e IX=DE. La matriz se copia en el área de copia.

La dirección de la pantalla actual se adapta al modo correspondiente y la validez de la posición se comprueba por 16FF. Si regresa con el acarreo a 0, significando que la posición no es válida, desembocará en una acción correctiva.

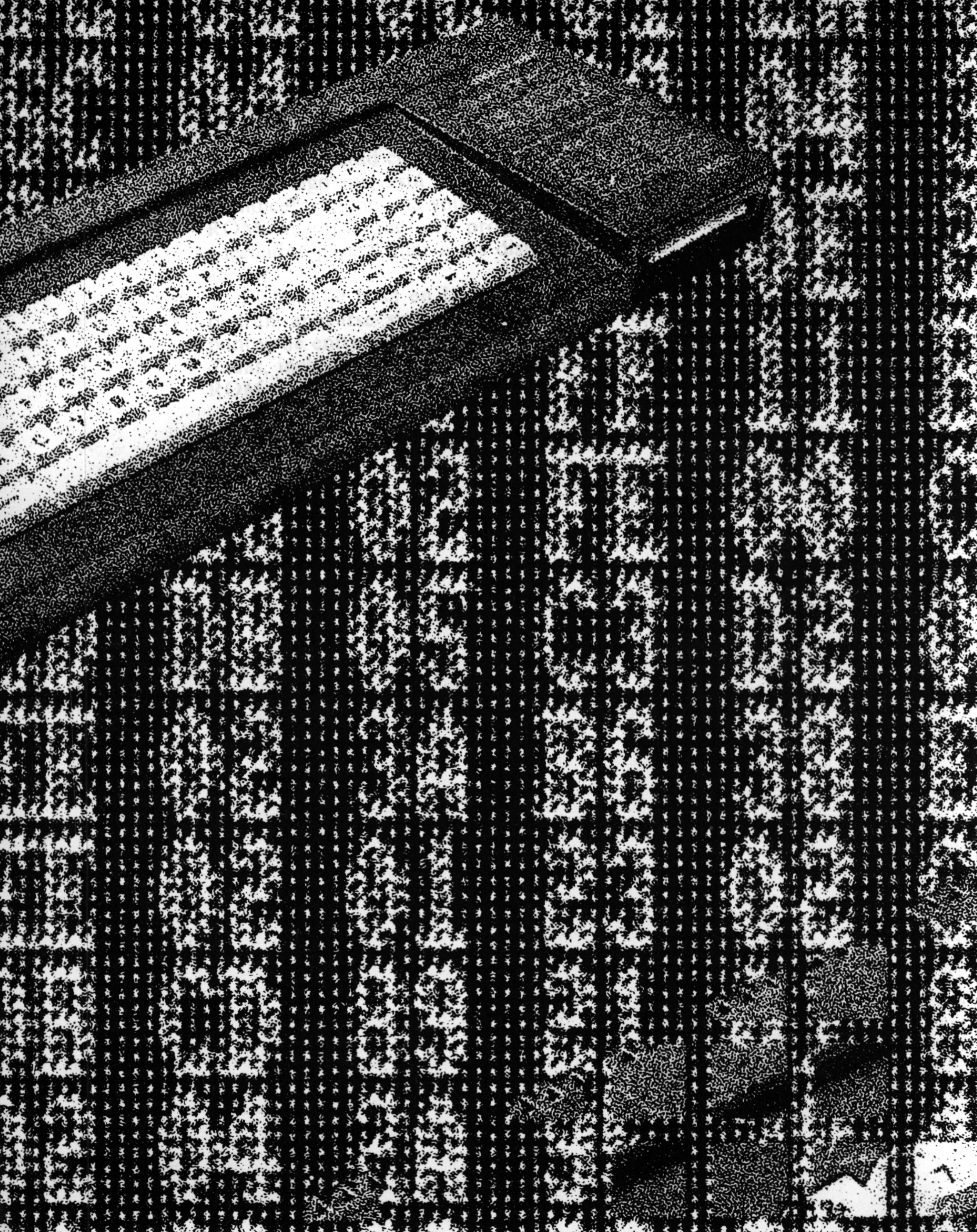
Lo que sigue es similar a la acción de TXT ESCRIBE CARACTER, con la salvedad de que se utilizan coordenadas de gráficos. La rutina sale vía GRA MOV ABSOLUTO para colocar el cursor en la esquina superior izquierda de la posición del siguiente carácter.

Comentarios

El hecho de agrupar algunas de las funciones presenta un sistema de gráficos de apariencia relativamente simple, pero en realidad esconde mucha complejidad. La ausencia de funciones de creación de círculos y de relleno es de lamentar en algunas aplicaciones, pero existen programas que añaden dichas facilidades para aquellos que las necesiten.

Una cuestión que a veces se olvida es que el número de puntos comprendidos en un ancho de pantalla es 640 para el modo 2, 320 para el modo 1, y sólo 160 para el modo 0. Esto puede provocar contratiempos al intentar combinar modelos intrincados con un amplio espectro de colores.

Algunos programadores, especialmente algunos españoles, han hallado medios para crear figuras de alta complejidad con un uso aparentemente pródigo de colores; si tienes la oportunidad de estudiar sus métodos, te resultarán muy reveladores. A falta de ello, merece la pena probar varios experimentos, guiándose por los detalles aquí proporcionados.



9
El monitor de teclado

El monitor de teclado ocupa el área 19E0-1E62 de la ROM y utiliza como espacio de trabajo la zona B34C-B548. Hay 26 puntos de entrada definidos y una indirección.

El sistema utiliza tres "mapas de bits" para registrar las acciones del teclado, que se resumen en una tabla al final de esta sección. Cada bit individual está relacionado con una tecla determinada y los mapas se actualizan 50 veces por segundo, a través de una rutina llamada por el controlador de interrupciones. Esto significa que el contenido del mapa sólo puede examinarse con resultados coherentes mientras estén inhibidas las interrupciones. Observa que las interrupciones estarán siempre que el sistema de cassette esté en acción, por lo que el teclado aparece efectivamente "muerto" durante este período.

Cuando se pulsa una tecla, su correspondiente bit en el mapa 1 se pone a 0. El bit equivalente del mapa 2 es puesto a 1 si el bit del mapa 1 se modifica de 0 a 1, o es igual a cero. Esto mantiene el bit del mapa 2 puesto a 1 si el bit del mapa 1 varía debido a los rebotes en la pulsación de la tecla.

Entonces se examina el mapa 3. Si presenta 0 para una tecla en particular, y el mapa 2 asigna 1 a esa tecla, los datos se disponen en el área de memoria del teclado y el byte del mapa 3 se pone a 1 desde el byte del mapa 2.

Este proceso se aplica para todos los bytes de los 10 mapas, uno por

uno, tomando los bits de cada byte en orden creciente, de forma que sea posible hacer más de una entrada durante un solo examen del teclado, teniendo en cuenta que las entradas se hacen en orden ascendente de números de teclas. (Si examinaras el código pertinente, quedarías asombrado por la función B=A AND -A: coloca un bit en B, correspondiente al bit menos significativo de A que esté a uno.)

Los datos de la memoria del teclado no tienen nada que ver con los códigos ASCII y sólo una pequeña relación indirecta con el número de la tecla. Cada entrada en el *buffer* ocupa 2 bytes y el espacio disponible es para 40 bytes, de forma que se pueden almacenar 20 pulsaciones de tecla simultáneamente. El byte alto es una máscara de un solo bit, que indica de qué bit de un byte del mapa se trata. El byte bajo estará compuesto. Si se pulsa la tecla SHIFT, el bit 5 se pone a uno, y si se pulsa la tecla CONTROL, será el bit 7 el que se ponga a 1. Se añade el número del byte del mapa comprendido entre 0 y 9. Todos los datos necesarios están disponibles, pero hay que decodificarlos.

A modo de ilustración, examina qué pasaría al pulsar la tecla "@". Es la tecla 26=3*8+2, luego estamos tratando con el bit 2 del byte del tercer mapa. Si no hemos pulsado ni SHIFT ni CONTROL, la entrada a la memoria del teclado será 0403.

La memoria del teclado es del tipo "circular", utilizando dos punteros. El puntero de entrada está en (B53D), y el de salida, en (B53F). Son punteros que desplazan un solo byte. Cuando se añade una entrada al *buffer*, el puntero de entrada se incrementa, y cuando una entrada se elimina, el puntero de salida se incrementa. Cuando algún puntero alcanza el valor &14, se vuelve a poner a 0. Considera los punteros como si fuesen las manecillas de un reloj, persiguiéndose la una a la otra alrededor de la esfera, quedando el *buffer* utilizado comprendido entre las dos.

(B53C) se inicializa a &15. Se decrementa, cuando se intenta ejecutar una entrada al *buffer*; se incrementa, cuando lo que se intenta es eliminar una entrada. Si del decremento resulta 0, se elimina el intento de ejecutar una entrada, ya que el *buffer* está lleno, y (B53C) se reinicializa a 1. De esta forma contiene el espacio disponible del *buffer* (en palabras) más uno.

(B53E) se inicializa a 1 y se incrementa cuando se ejecuta una nueva entrada, decrementándose cuando se elimina. Si del decremento resulta 0, el *buffer* está vacío. Se abandona la intención de leer una entrada y (B53E) se pone a 1. Por tanto, contendrá el número de entradas del *buffer* (en palabras) más uno.

(B540) se inicializa a 0 y se incrementa cuando se ejecuta una entrada, decrementándose cuando una entrada se elimina. Por consiguiente, ofrecerá una comprobación del número actual de entradas al *buffer*, dato que puede ser de utilidad.

Las direcciones de acceso a la memoria se calculan añadiendo dos veces los punteros a B514. Puesto que (B53D) y (B53F) trabajan más allá del intervalo 0 a &13, el *buffer* ocupa el área B514-B53B.

Un punto importante a tener en cuenta es que los estados de SHIFT

almacenados en la memoria se determinan desde los bits 5 y 7 del byte bajo de la entrada al *buffer*, que dependen del estado de SHIFT en el momento de ejecutar la entrada. Los estados de SHIFT en el momento de leer la entrada son irrelevantes. Sin embargo, los estados de SHIFT-LOCK deben implementarse por separado. Observa que la información referente a cambios de estado se intercala con otras entradas del teclado.

Rutinas de teclado

La descripción general anterior del sistema de teclado ha traído a la luz algunos puntos importantes referentes a su funcionamiento, tales como la longitud del *buffer* de teclado; sin embargo, se puede utilizar el sistema sin preocuparse demasiado por su forma de actuar. Pero la comprensión del sistema es de gran ayuda para explicar cómo funcionan las llamadas del sistema.

Será mejor comenzar con dos rutinas que restauran el sistema en un estado estándar, que pueden ser valiosas si estás tentado a experimentar y obtienes resultados inesperados o indeseados.

TCL INICIALIZACION *(KM INITIALISE):* BB00, 19E0

Esta es la función principal de restauración que afecta a todos los elementos:

- Las tablas de tecla/código y parte de la tabla de control de repeticiones se restauran copiando (1D69)-(1E62) en (B34C)-(B445) (véase tabla).
- Los estados del teclado CAPS y SHIFT-LOCK en (B4E7/8) se ponen a cero.
- La velocidad de repetición en (B4E9) se pone a 2.
- La pausa de repetición en (B4EA) se pone a &1E.
- El mapa 2, (B45F)-(B4FE) se dispone para &FF entradas.
- El mapa 3, en (B4EB)-(B4F4) se dispone para cero entradas.
- (B541/2) se pone a B34C, base de la tabla 1 de tecla/código.
- (B543/4) se pone a B39C, base de la tabla 2 de tecla/código.
- (B545/6) se pone a B43C, base de la tabla de control de repetición.
- La rutina sale vía TCL RE-INICIALIZACION.

TCL RE-INICIALIZACION *(KM RESET):* BB03, 1A1E

Esta rutina de inferior grado de restauración es, sin embargo, bastante enérgica.

- (B53C)=&15 (espacio libre de *buffer* más uno).
- (B53D)=0 (puntero de entrada al *buffer*).
- (B53E)=1 (entradas del *buffer* más uno).
- (B53F)=0 (puntero de salida del *buffer*).
- (B440)=0 (número de entradas al *buffer*).
- (B4E0)=&FF ("ignora" el carácter devuelto).

Se asigna un *buffer* de cadena de teclas de 152 bytes (&98) comenzando en B446, y la parte vacía de la cadena se rellena hasta completarla (por desgracia, ¡esto se superpone a la tabla de control de repetición!).

Se acude a la indirección TCL EXAMINA ESC en BDEE para las direcciones omitidas.

La rutina sale vía NC INHIBE RUPTURAS.

Rutinas de entrada

El hecho de que haya cuatro rutinas fundamentales de entrada de datos puede resultar desconcertante, pero su interrelación es importante.

TCL LEE TECLA *(KM READ KEY):* BB1B, 1B5C

Si hay una entrada en el *buffer* del teclado, la rutina sale con el código apropiado en A, poniéndose a 1 el acarreo. Si la memoria está vacía, la rutina regresa con el acarreo a cero. Todos los registros, salvo AF, son preservados.

TCL ESPERA TECLA *(KM WAIT KEY):* BB18, 1B56

Se llama a TCL LEE TECLA repetidamente, hasta que regrese con el acarreo puesto a uno. Esto se puede utilizar con "pulse cualquier tecla para continuar", puesto que TCL LEE TECLA sólo examina la memoria para ver si se ha pulsado alguna tecla desde la última comprobación. Todos los registros son preservados, salvo AF.

TCL LEE CARACTER *(KM READ CHAR):* BB09, 1A42

En primer lugar se comprueba el carácter "devuelto" en (B4E0). Si no es &FF, el carácter es devuelto a A con el acarreo puesto a 1 y (B4E0)=&FF (véase TCL REGRESA CARACTER).

Después se comprueba (B4DE/F). Si (B4DF)≠0, se ha sacado parcialmente una cadena de teclas y la rutina regresa con el código para el siguiente carácter de la cadena cargado en A, con el acarreo puesto a 1.

Si no hay carácter devuelto ni cadena de teclas, TCL LEE TECLA es llamada para buscar una entrada al *buffer* del teclado. Si el código devuelto a A es un símbolo de cadena de teclas, se inicia la salida de la cadena; en caso contrario, la rutina regresa con el acarreo puesto a 1 y el código en A.

Si no hay carácter devuelto, carácter de cadena de teclas, ni código de memoria, la rutina regresa con el acarreo a cero.

En cualquier caso, todos los registros, salvo AF, son preservados.

TCL ESPERA CARACTER *(KM WAIT CHAR):* BB06, 1A3C

Se llama a TCL LEE CARACTER repetidamente, hasta que regrese con el acarreo a uno.

Las diferencias entre estas rutinas deben estar ya claras. Las versiones de ESPERA se enlazan hasta encontrar un código, puesto que las formas de LEE dan un rápido vistazo. Sólo las versiones de CARACTER examinan las devoluciones y las cadenas.

TCL REGRESA CARACTER *(KM CHAR RETURN):* BB0C, 1A77

Con esta rutina tú te lo guisas y tú te lo comes. Cuando un carácter ha sido leído, ha sido tomado del *buffer* del teclado y ya no sirve más. Llamando a TCL REGRESA CARACTER se transfiere el código de A a (B4E0), por lo que puede leerse por TCL LEE CARACTER como si acabara de salir del *buffer* del teclado. Sólo puede “devolverse” un carácter cada vez, puesto que sólo se dispone de una posición.

Todos los registros son preservados.

Cadenas de teclas

La memoria estándar de cadenas de teclas se encuentra en B446-B4DD, que se superpone a la tabla de control de repetición, pero puede definirse una memoria alternativa en cualquier lugar de la RAM. De las

152 posiciones de la memoria estándar, 49 se cargan para cubrir lagunas y quedan 103 para otras definiciones.

El formato del *buffer* es simple. Cada cadena va precedida por su longitud en bytes, de forma que la cadena "n" puede hallarse saltando n veces desde un byte de longitud hasta el siguiente. Las restantes posiciones se rellenan de la siguiente forma:

```
B446                           01 30 01 31 01 32 01 33 01 34
B450  01 35 01 36 01 37 01 38 01 39 01 2E 01 0D 05 52
B460  55 4E 22 0D
```

Las 10 primeras cadenas tienen una longitud de un carácter y proporcionan códigos para los números del 0 al 9. La undécima cadena da un punto (&2E), la duodécima el código ENTER, mientras que la decimotercera cadena tiene 5 bytes que son "RUN", seguido de ENTER.

Puede construirse un *buffer* de usuario, con las anteriores cadenas supletorias previamente cargadas, utilizando:

TCL ASIGNA BUFFER *(KM EXP BUFFER):* BB15, 1A7B

A la entrada, DE debe contener la dirección de comienzo del nuevo *buffer*, HL debe contener su longitud, que deberá ser &31 o más (aunque no 44, que está establecida en la documentación oficial).

Si el *buffer* se ha establecido, la rutina regresa con el acarreo a uno, y si ha habido algún fallo, el acarreo estará a cero.

En el caso de que una cadena esté en proceso de salida, (B4DF) se pone a cero, haciendo que la extracción se detenga inmediatamente.

El siguiente paso es disponer las cadenas que desees. Esto se realiza por:

TCL PONE EXPANSION *(KM SET EXPAND):* BB0F, 1ABD

A la entrada, B debe contener la clave de expansión relevante, C debe contener la longitud de la cadena y HL debe apuntar a la fuente de la nueva cadena. Si A regresa con el acarreo a uno, es que ha habido suerte; y si está puesto a cero, es que no se ha conseguido, quizá porque la memoria no fuera lo suficientemente grande como para contener la cadena.

En primer lugar se determina la posición de la cadena en la memoria, utilizando una rutina que dispone un puntero en HL, inicialmente al comienzo del *buffer;* lee la longitud del byte; añade esta longitud más uno a HL; utiliza este resultado para leer la longitud del siguiente byte, y así sucesivamente. Cuando este proceso se ha realizado un número de veces igual a la longitud de la clave menos &80, HL apunta a la posición de la cadena. Todas las entradas anteriores son entonces desplazadas para dejar el espacio suficiente para la siguiente cadena, que puede ser copiada entonces en su sitio. (Observa que el movimiento puede ser hacia arriba o hacia abajo, según que la nueva cadena sea más larga o más corta que la cadena a la que sustituye.)

Comparado con algunas implementaciones de cadenas de teclas, este procedimiento es maravillosamente simple.

Para leer una cadena se necesita:

TCL TOMA EXPANSION *(KM GET EXPAND):* BB12, 1B2E

A la entrada, A debe contener la señal de expansión, mientras que L contiene el número del carácter dentro de la cadena que se va a leer. A la salida, el éxito se traduce mediante el acarreo puesto a uno, con el código del carácter en A. Si el proceso de extracción de cadena se ha completado, el acarreo se pone a cero y A se altera. En ambos casos, DE se altera.

Esta llamada se utiliza por TCL LEE CARACTER, y normalmente no se empleará de forma independiente, puesto que requiere asociarse a una rutina que actualice los *flags* adecuados y el puntero L.

Para saber si se ha pulsado una tecla determinada, sin identificar su código, se puede utilizar:

TCL EXAMINA TECLA *(KM TEST KEY):* BB1E, 1CBD

A la entrada, A debe contener un número de tecla. Si al regreso A=0, significará que la tecla está pulsada. C contiene los estados actuales de SHIFT y CONTROL (véase a continuación) y los estados LOCK son ignorados. A y HL son alterados.

La información suministrada procede del mapa 3. Los bits de SHIFT y CONTROL son aislados en (B4ED) y se cargan en C. El número de tecla se convierte entonces en punteros de byte y bit y se utiliza para aislar el bit relacionado con la tecla especificada. Si la tecla se ha pulsado, el bit es 1.

El estado de CAPS-LOCK y SHIFT-LOCK puede comprobarse mediante:

TCL TOMA ESTADO *(KM GET STATE):* BB21, 1BB3

Al regresar, esta rutina dispone L=&FF si es efectiva la función SHIFT-LOCK, y H=&FF si es efectiva la función CAPS-LOCK; en todo caso, los registros contienen cero. Todos los demás registros son preservados, puesto que la rutina tan sólo coloca HL=(B4E7/8), posiciones que se alternan cuando se leen los códigos LOCK adecuados desde la memoria del teclado.

Completando los procesos de lectura del teclado:

TCL ESTADO JOYSTICK *(KM GET JOYSTICK):* BB24, 1C5C

A la salida, L=(B4F1) AND &7F, dando los datos para el *joystick* 1 (teclas 48-54) y H=(B4F4 AND &7F), dando los datos para el *joystick* 0 (teclas 72-78). A=H. Todos los demás registros son preservados.

Tablas de teclas y códigos

La relación entre las teclas y los códigos que éstas generan es determinada por las tres tablas de teclas situadas en el área B34C-B43B (véase tabla). Puesto que las tablas están en la RAM, cualquier tecla puede ser designada para producir cualquier código deseado, con la limitación de que los códigos del intervalo &80-&9F serán tratados como señales de cadenas por TCL LEE CARACTER (pero no por TCL LEE TECLA).

Hay tres llamadas disponibles para cambiar las entradas a la tabla de teclas:

TCL PON TRASLADO *(KM SET TRANSLATE):* BB27, 1D52 Tabla 1

TCL TRASLADO CON SHIFT *(KM SET SHIFT):* BB2D, 1D57 Tabla 2

TCL TRASLADO CON CONTROL *(KM SET CONTROL):* BB33, 1D5C Tabla 3

A la entrada, A debe contener un número de tecla y B el código que va a generar dicha tecla. Una vez haya sido leída la base de la tabla apropiada, la rutina es común para los tres puntos de entrada. Si A excede de &4F, la rutina es devuelta, regresando con el *flag* C a cero. En otro caso, A se añade a la base de la tabla para formar un puntero que sitúe el contenido de B en la tabla. La tabla 1 se relaciona con las entradas sin SHIFT y sin CONTROL. La tabla 2 se utiliza cuando la función CONTROL es efectiva.

Una correspondiente colocación de las entradas permite leer un código mediante:

TCL TOMA TRASLADO *(KM GET TRANSLATE):* BB2A, 1D3E Tabla 1

TCL OBTIENE SHIFT *(KM GET SHIFT):* BB30, 1D43 Tabla 2

TCL OBTIENE CONTROL *(KM GET CONTROL):* BB36, 1D48 Tabla 3

El formato es muy similar al anterior para las tres llamadas previas, salvo que el número de tecla contenido en A a la entrada se utiliza para leer un byte desde la tabla apropiada y devolverlo a A. HL es alterado.

El estado LOCK del teclado es ignorado, tomándose en cuenta por TCL LEE TECLA.

Acción repetitiva

La acción repetitiva de una tecla puede modificarse nominalmente mediante:

TCL IMPONE REPETICION *(KM SET REPEAT):* BB39, 1CAB

A la entrada, A contiene un número de tecla. Si B contiene 0, la tecla podrá repetir, mientras que si B contiene &FF se impedirá la repetición. Un número de tecla mayor que &4F es rechazado, y para los demás, B se copia en la tabla de control de repeticiones.

El problema es que, si A excede de 9, B también se copiará en el *buffer* estándar de cadena-tecla, provocando un caos, a menos que hayamos dispuesto un *buffer* diferente por nuestra cuenta...

Para leer la tabla de repetición:

TCL CONOCE REPETICION *(KM GET REPEAT):* BB3C, 1CA6

A la entrada, A contiene un número de tecla. Si esta tecla puede repetirse, la rutina regresa con Z a cero. A y HL se alteran.

Las repeticiones constantes se manejan con:

TCL IMPONE RETARDO *(KM SET DELAY):* BB3F, 1C6D

A la entrada, H debe contener el factor de retardo y L el factor de velocidad. A falta de otros, los valores son &1E, que proporcionan un retardo de 30/50=0,6 segundos, y L=2, que produce una velocidad de 50/2 repeticiones por segundo. La rutina regresa con AF alterado.

TCL CONOCE RETARDO *(KM GET DELAY):* BB42, 1C69

El factor de retardo es devuelto en H y el factor de velocidad en L. AF es altera.

Funciones de ruptura

Para entender las llamadas de ruptura, es necesario haber leído previamente la sección sobre sucesos.

TCL INHIBE RUPTURAS *(KM DISARM BREAK):* BB48, 1C82

(B50C) se pone a cero para marcar la condición inhibida. NC BORRA SUCESO es llamada con HL = B50D para trasladar el suceso de ruptura desde la lista de síncronos. La rutina regresa con AF y HL alterados.

TCL PERMITE RUPTURAS *(KM ARM BREAK):* BB45, 1C71

A la entrada, DE contiene C45E, la dirección de la rutina del suceso de ruptura, y C contiene el número adecuado de la ROM. (&FD, que significa ROM no modificada, activa la ROM superior e inhibe la inferior.) Con esto se accede a la rutina de ruptura en el intérprete de BASIC, pero pueden utilizarse otros datos de entrada para acceder a una rutina alternativa.

TCL INHIBE RUPTURAS es llamada para establecer un estado conocido. Entonces se llama a NC INICIALIZA SUCESO con HL = B50D y B = &40 para inicializar un bloque de sucesos en B50D-B513. La clase es "dirección lejana", "urgente", "síncrono". La dirección de la rutina y la ROM son las especificadas en DE y C. (B50C) = &FF señala la condición permitida.

TCL GENERA RUPTURA *(KM BREAK EVENT):* BB4B, 1C90

Si (B50C) = 0, la rutina regresa. En otro caso, (B50C) = 0 y NC SUCESO es llamada con HL = B50D para atender al suceso de ruptura. &EF se coloca en la memoria del teclado para señalar el punto en el que el suceso fue atendido.

TCL EXAMINA ESC *(KM TEST BREAK):* BDEE, 1C90

Al entrar en esta indirección, la interrupción debe ser desactivada para inhibir la acción del teclado, y la ROM inferior debe ser activada. C debe contener el estado de la tecla SHIFT/CONTROL, como se encontró después de inhibir la interrupción.

Si SHIFT y CONTROL no están pulsadas simultáneamente, la rutina saldrá vía TCL GENERA RUPTURA.

En otro caso, parece que se está pidiendo un *reset*, pero para asegurarse los bytes se añaden al mapa 3. Si únicamente se han pulsado las teclas SHIFT, CONTROL y ESCAPE, el total debería ser &A4, y si se consigue este resultado, se producirá a continuación un salto a 0000. En otro caso, la rutina saldrá vía TCL GENERA RUPTURA.

Sumario

Realmente se puede hacer muy poco con un teclado, aparte de solicitar que proporcione datos. La mayoría de las llamadas listadas anteriormente resultan apropiadas sobre todo para otras llamadas que utilicen los datos obtenidos. Sin embargo, hay algunos trucos que merece la pena mencionar.

Se puede vaciar la memoria del teclado variando los punteros, pero deben modificarse todos ellos. Hay una rutina (en 1CED en la versión 1.0) que realiza exactamente esto. (Es el destino de la primera llamada en TCL RE-INICIALIZACION, por lo que será fácil encontrarla en otras versiones.)

Resulta útil colocar una memoria alternativa tecla-cadena, para que pueda trabajar la tabla de repetición.

Puedes disponer una tabla especial tecla/código por tu cuenta, poniéndola en acción cuando sea necesario. Esto te proporcionará códigos extra que serán generados mediante la acción del teclado.

Una consideración particularmente importante se impone si utilizas un programa primario que concierna al tema de las rupturas. La dirección C45E sirve para el intérprete de BASIC, pero puede que no te sirva en absoluto para tu programa; sin embargo, si utilizas una ROM lateral, entrará en C45E como respuesta al pulsar ESCAPE.

En algunos aspectos, el monitor de teclado es una de las secciones menos satisfactorias del sistema operativo, pero funciona razonablemente bien, a pesar de todo. Mientras conozcas sus peculiaridades, no hay motivos para preocuparte.

Mapas de bits

Mapa 1	Mapa 2	Mapa 3	Bit Ø	1	2	3	4	5	6	7
B4F5	B4FF	B4EB	Ø	1	2	3	4	5	6	7
B4F6	B5ØØ	B4EC	8	9	1Ø	11	12	13	14	15
B4F7	B5Ø1	B4ED	16	17	18	19	2Ø	21	22	23
B4F8	B5Ø2	B4EE	24	25	26	27	28	29	3Ø	31
B4F9	B5Ø3	B4EF	32	33	34	35	36	37	38	39
B4FA	B5Ø4	B4FØ	4Ø	41	42	43	44	45	46	47
B4FB	B5Ø5	B4F1	48	49	5Ø	51	52	53	54	55
B4FC	B5Ø6	B4F2	56	57	58	59	6Ø	61	62	63
B4FD	B5Ø7	B4F3	54	65	66	67	68	69	7Ø	71
B4FE	B5Ø8	B4F4	72	73	74	75	76	77	78	79

Los números de la tabla son números de teclas.

Tablas tecla/código

N.º de tecla	Ø	1	2	3	4	5	6	7	8	9	1Ø	11	12	13	14	15
Tabla 1	FØ	F3	F1	89	86	83	8B	8A	F2	EØ	87	88	85	81	82	8Ø
Tabla 2	F4	F7	F5	89	86	83	8B	8A	F6	EØ	B7	88	85	81	82	8Ø
Tabla 3	F8	FB	F9	89	86	83	8C	8A	FA	EØ	87	88	85	81	82	8Ø

N.º de tecla	16	17	18	19	2Ø	21	22	23	24	25	26	27	28	29	3Ø	31
Tabla 1	1Ø	5B	ØD	5D	84	FF	5C	FF	5E	2D	4Ø	7Ø	3B	3A	2F	2E
Tabla 2	1Ø	7B	ØD	7D	84	FF	6Ø	FA	A3	3D	7C	5Ø	2B	3A	3F	3E
Tabla 3	1Ø	1B	ØD	1D	84	FF	1C	FF	1E	FF	ØØ	1Ø	FF	FF	FF	FF

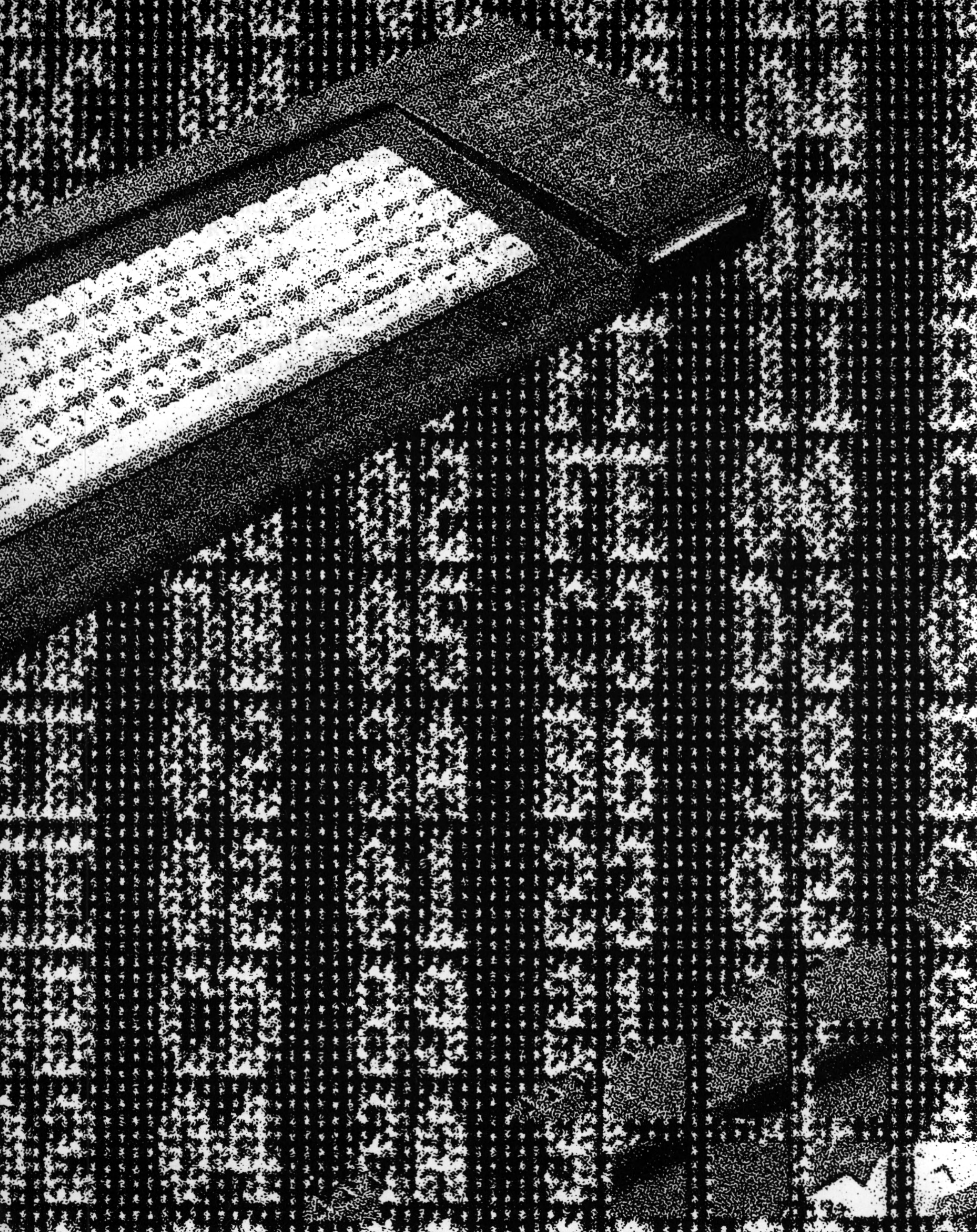
N.º de tecla	32	33	34	35	36	37	38	39	4Ø	41	42	43	44	45	46	47
Tabla 1	3Ø	39	6F	69	6C	6B	6D	2C	38	37	75	79	68	6A	6E	2Ø
Tabla 2	5F	29	4F	49	4C	4B	4D	3C	28	27	55	59	48	4A	4E	2Ø
Tabla 3	1F	FF	ØF	Ø9	ØC	ØB	ØD	FF	FF	FF	15	19	Ø8	ØA	ØE	FF

N.º de tecla	48	49	5Ø	51	52	53	54	55	56	57	58	59	6Ø	61	62	63
Tabla 1	36	35	72	74	67	66	62	76	34	33	65	77	73	64	63	78
Tabla 2	26	25	52	54	47	46	42	56	24	23	45	57	53	44	43	58
Tabla 3	FF	FF	12	14	Ø7	Ø6	Ø2	16	FF	FF	Ø5	17	13	Ø4	Ø3	18

N.º de tecla	64	65	66	67	68	69	7Ø	71	72	73	74	75	76	77	78	79
Tabla 1	31	32	FC	71	Ø9	61	FD	7A	ØB	ØA	Ø8	Ø9	58	5A	FF	7F
Tabla 2	21	22	FC	51	Ø9	41	FD	5A	ØB	ØA	Ø8	Ø9	58	5A	FF	7F
Tabla 3	FF	7E	FC	11	E1	Ø1	FE	1A	FF	FF	FF	FF	FF	FF	FF	7F

Espacio de trabajo del monitor de teclado

B34C-B39B	Tabla 1 de código/tecla: tecla SHIFT sin pulsar
B39C-B3EB	Tabla 2 de código/tecla: tecla SHIFT pulsada
B3EC-B43B	Tabla 3 de código/tecla: tecla CONTROL pulsada
B34C-B49B	Tabla de control de repeticiones
B446-B4DD	*Buffer* del teclado (cuidado con los solapamientos)
B4DE	Puntero del *buffer*
B4DF	Señal de la cadena actual
B4E0	Devuelve carácter
B4E1/2	Comienzo del *buffer* de cadenas
B4E3/4	Final del *buffer* de cadenas
B4E5/6	Comienzo del espacio libre en el *buffer* de cadenas
B4E7/8	Estado LOCK del teclado
B4E9	Velocidad de repetición
B4EA	Pausa o retardo de repetición
B4EB-B4F4	Mapa 3
B4F5-B4FE	Mapa 1
B4FF-B508	Mapa 2
B509	Cuenta de repetición
B50A	Número del mapa de bytes
B50B	Mapa de máscaras de bit
B50C	*Flag* para permitir rupturas
B50D-B513	Bloque de ruptura de sucesos
B514-B53B	*Buffer* del teclado
B53C	Espacio libre del *buffer*+1
B53D	Puntero de entrada
B53E	Entradas del *buffer*+1
B53F	Puntero de salida
B540	Entradas del *buffer*
B541/2	Puntero de la tabla 1 de la tecla/código
B543/4	Puntero de la tabla 2 de la tecla/código
B545/6	Puntero de la tabla 3 de la tecla/código
B547/8	Puntero de la tabla de control de repeticiones



10 El monitor de cassette

El monitor de cassette ocupa el área 2370-2A91 de la ROM inferior y utiliza como espacio de trabajo la zona B800-B8D4 de la RAM. Añadiéndose a esta zona, se emplearán áreas de 2 Kbytes para almacenamiento temporal durante las entradas y salidas. Hay definidas 22 entradas o rutinas de *firmware*.

Las grabaciones están basadas en ciclos de ondas cuadradas. Los ciclos que representan bits en nivel alto tendrán doble duración que los que representan bits en nivel bajo. La frecuencia principal nominal puede variar entre 700 y 2500 Hz, siendo la frecuencia para omisiones de 1000 Hz, la cual le da al ciclo de un bit a nivel bajo una duración de 666 microsegundos.

La duración de un ciclo que representa a un bit a nivel alto será de 1332 microsegundos.

La precompensación se utiliza para desequilibrar la duración proporcional de las ondas cuadradas, siendo el valor prescrito para omisiones el de 25 microsegundos. Este valor se "añade" al ciclo del bit a nivel alto y se resta del ciclo del bit a nivel bajo, enfatizándose así la diferencia entre ambos. La precompensación se incrementará según se incremente la frecuencia principal.

Una faceta bastante útil del sistema es que éste se ajusta automáticamente a la frecuencia correcta durante la lectura de la cinta. Esto se basa en la lectura del bloque de cabeza, que está construido de esta forma:

- Zona ante-grabación *(gap)*.
- 2048 ciclos de bit a nivel alto.
- Un ciclo de bit a nivel bajo.
- Un byte de sincronismo.

Para datos, el byte de sincronismo es &16; para un encabezamiento es &2C.

La estructura en conjunto de un fichero consiste en un encabezamiento seguido por una grabación de datos. El bloque de encabezamiento consta de 64 bytes, pero la mayoría de ellos no tiene una función asignada. Durante la grabación, el bloque de encabezamiento se toma del área B807-B846. Durante la lectura, el encabezamiento se almacena en el área B88C-B8CB. Por referencia del encabezamiento durante la lectura, el sistema puede decidir si cargar el bloque de datos correspondiente o emitir el mensaje *found* (encontrado), que permitirá recuperarse de un error en la lectura, rebobinando la cinta y pidiendo nuevamente la lectura por si alguno de los bloques ha sido omitido.

El sistema es francamente tolerante con los lloros y fluctuaciones de la cinta, aunque puede fallar en casos extremos.

Los bloques de datos podrían ser de hasta 65536 bytes cada uno, pero es más corriente limitarlos a 2048 bytes, para mantener el tamaño del *buffer* dentro de lo razonable. Los bloques se dividen en segmentos de 258 bytes cada uno, utilizándose dos de esos bytes para comprobación cíclica del sincronismo (CCS).

Como los procesos de grabación y reproducción deben ser continuos, el sistema de interrupciones queda inhibido mientras la grabadora está en uso. Por la misma razón, los bloques construidos a partir de bytes creados a intervalos, o los que deben leerse en el modo byte a byte, son manipulados por medio de un *buffer* que actúa como intermediario. Durante el proceso de grabación, el *buffer* se actualiza según vayan estando los bytes disponibles. Cuando el *buffer* esté lleno, su contenido será grabado en la cinta. Esta última acción también se llevará a cabo si se cierra el *buffer*. Análogamente, en la reproducción del *buffer* se llenará inicialmente con un bloque completo y se volverá a llenar en el momento en que todos los datos hayan sido tomados por el procesador.

Una excepción a esto se produce cuando un bloque de datos, tal como un programa BASIC, ya está establecido, y por ello puede ser grabado directamente. Lo mismo es aplicable a la reproducción de datos de este tipo. El *buffer* intermedio sigue utilizándose, pero las transferencias a y desde la cinta son automáticas.

Las llamadas para el método de *buffer* intermedio son CAS LEE CARACTER y CAS SALIDA CARACTER, mientras que para el método directo las llamadas son CAS ENTRADA DIRECTA y CAS SALIDA DIRECTA.

Algunas de las llamadas ofrecen facilidades no disponibles desde el BASIC. CAS COMPARA nos provee de una fácil verificación, mientras

que CAS MENSAJES nos permite suprimir los mensajes de ayuda. CAS RETORNO reinicializa los punteros del *buffer* intermedio, de forma que una entrada pueda ser leída más de una vez. Por último, las llamadas de ABANDONO nos permiten desechar el contenido del *buffer*.

Para aquellos a quienes se les antoje experimentar con sistemas de grabación por sí mismos, tal vez buscando la compatibilidad con sistemas de grabación de otros ordenadores, están accesibles las primitivas rutinas CAS LEE y CAS ESCRIBE. Estas dos rutinas tratan con grabaciones sencillas, manipulando datos de una longitud dada, desde o hacia una posición inicial dada de la memoria.

El sistema de temporización implica el uso del registro de refresco del Z-80; una aplicación poco usual de él, que dará una temporización corta pero muy precisa.

Mensajes

Los mensajes de ayuda están almacenados de forma comprimida, con los espacios omitidos pero recurridos, añadiendo el bit 7 del código del carácter precedente. Hay cuatro avisos de error principales:

Read Error a	Bit demasiado largo.
Read Error b	Fallo en la comprobación de sincronismo (CCS).
Read Error c	Bloque demasiado largo.
Write Error a	Frecuencia demasiado alta.

El error a de lectura puede ocurrir si la cinta ha sido parada durante la reproducción. También ha sido visto en un caso extremo de lloros, que frenaban la marcha de la cinta de cuando en cuando. El error b de lectura es más común, e implica un defecto en la superficie de la cinta. Los otros errores no han sido observados, pero podían inducirse deliberadamente.

La tecla ESCAPE ofrece una salida de las rutinas de cassette sólo en limitadas circunstancias. Esta tecla se detecta directamente, no a través del *buffer* del teclado, que es inefectivo en ausencia de interrupciones. Esto significa que puede ser necesario mantener la tecla presionada un cierto tiempo antes de que surta efecto.

Los nombres de los ficheros pueden contener hasta 16 caracteres. Todos los caracteres adicionales serán ignorados, mientras que los nombres más cortos se rellenarán con espacios hasta los 16 bytes.

CAS INICIALIZA *(CAS INITIALISE):* BC65, 2370

Se efectúa una llamada a CAS ABANDONA ENTRADA y otra a CAS ABANDONA SALIDA. Después, con A=0 se llama a CAS MENSAJES para activar los mensajes de ayuda de pantalla. Luego se llama a CAS VELOCIDAD con HL=014D (333) y A=&19 (25) para fijar la velocidad en caso de omisión y el valor de la precompensación.

CAS VELOCIDAD *(CAS SET SPEED):* BC68, 237F

A su entrada, HL debe contener la mitad de la longitud del ciclo para niveles bajos, expresada en microsegundos, y A debe contener la precompensación requerida, también en microsegundos. Se multiplica HL por 64. A se divide entre 4 y se suma a HL. El resultado se guardará en (B8D1/2).

BC y DE no se ven alterados.

CAS MENSAJES *(CAS NOISY):* BC6B, 238E

El contenido de A se guarda en (B800). Si A=0, las ayudas son habilitadas; en caso contrario, los mensajes de ayuda no aparecerán.

Todos los registros conservan su contenido.

CAS ARRANCA MOTOR *(CAS START MOTOR):* BC6E, 2A4B

No existen requisitos de entrada. Si la acción fue completada, la rutina regresará con el acarreo a uno, pero, si la tecla ESC se pulsó, el acarreo estará a cero. También al regreso, A contiene el estado previo del puerto C del PPI; el bit 4 de dicho puerto controla el estado del motor (véase CAS RESTAURA MOTOR).

Después de la puesta en marcha del motor, se produce un retardo de unos dos segundos antes del retorno de la rutina, para dar tiempo a que el motor adquiera la velocidad apropiada, si es que no estaba ya en funcionamiento desde antes.

CAS PARA MOTOR *(CAS STOP MOTOR):* BC71, 2A4F

Es idéntica a CAS ARRANCA MOTOR, excepto en que el motor se para en vez de arrancarse. En este caso no existe ningún retardo antes del regreso.

CAS RESTAURA MOTOR *(CAS RESTORE MOTOR):* BC74, 2A51

A su entrada, A debe contener el byte que contenía A al retorno de las dos rutinas anteriores. El estado anterior del motor pasa a ser el actual.

Estas tres rutinas de control del motor son invocadas por otras rutinas de nivel superior y únicamente tienen interés para el usuario si éste desea mover la cinta sin grabar o reproducir. Sin embargo, la tecla PLAY debe estar pulsada para permitir el movimiento. Como posible aplicación puede emplearse para encontrar automáticamente una grabación dada, haciendo funcionar el motor durante un tiempo ya calculado.

CAS ABRE ENTRADA *(CAS IN OPEN):* BC77, 2392

Esta rutina inicializa un *buffer* de entrada, etiquetado con el nombre del fichero dado, así como los procedimientos necesarios para llenar el *buffer* desde la cinta y hacer que los datos estén disponibles a la acción del programa.

A su entrada, B debe contener el número de bytes del nombre del fichero; HL contendrá la dirección de comienzo del nombre del fichero, y DE debe apuntar al área de 2 Kbytes, que se utilizará como *buffer*. Como la lectura de este *buffer* se hace desde LAM RAM, éste deberá permanecer fuera de los márgenes de la ROM.

Se inicializa entonces el área B802-B846, en su mayor parte por una rutina formada con CAS ABRE SALIDA. Los bytes no señalados se ponen a cero. El *buffer* se actualiza mediante los datos leídos de la cinta, siendo llamada automáticamente la acción de lectura.

Si todo va bien, la rutina regresará con el acarreo a uno y el *flag* Z a cero. HL apunta al encabezamiento del fichero recién almacenado; DE contiene la localización de los datos especificada en el encabezamiento, y A contiene el tipo de fichero.

Si un fichero de entrada ya había sido abierto, la rutina regresará con

C y Z a cero, mientras que un regreso con C a cero y Z a uno indicará que la tecla ESC fue pulsada.

Si la rutina ha sido ejecutada con éxito, podrán leerse del *buffer* hasta 2 Kbytes, mediante el empleo repetido de CAS LEE CARACTER.

CAS CIERRA ENTRADA *(CAS IN CLOSE):* BC7A, 23FC

CAS ABANDONA ENTRADA *(CAS IN ABANDON):* BC7D, 2401

La diferencia entre estas llamadas es que CIERRA ENTRADA regresará con C a cero y sin hacer nada, si no había ningún fichero de entrada abierto. En otro caso, acudirá a ABANDONA ENTRADA, que será ejecutada incondicionalmente.

No existen requisitos de entrada. Se pone a cero (B802) para marcar al fichero como cerrado. La dirección de comienzo del *buffer* se copia en DE desde (B803/4).

Se carga A con (B8CC) XOR 1, y si A=0, se pone a cero (B8CC). Este es el *flag* para los mensajes de ayuda. El contenido de DE puede utilizarse para reestablecer el *buffer*.

CAS LEE CARACTER *(CAS IN CHAR):* BC80, 2435

Esta rutina introduce en A el siguiente byte del *buffer* intermedio, previa comprobación de validez. No hay requisitos de entrada; toda la acción depende de punteros internos.

En caso de que (B802) contenga 1 ó 2, la rutina retornará inmediatamente con los *flags* C y Z a cero y sin que se lleve a cabo ninguna acción. 1 indica que el *buffer* está abierto, los datos aún no se han tomado. 2 indica que CAS LEE CARACTER ya ha sido utilizada una vez al menos. En otro caso, se hace que (B802)=2.

Si la cuenta de bytes disponibles que está en (B81A/B) es 0000, todos los datos se han tomado y se hace entonces un intento de leer en la cinta otra parte del fichero. Si se ha alcanzado el final de los datos del fichero, la rutina regresará con C y Z a cero.

Si el *buffer* todavía contiene datos, entonces se decrementa (B81A/B) y HL=(B805/6), que es el puntero del *buffer*. LAM RAM lee el siguiente byte,

lo introduce en A y se incrementa (B805/6). La salida de la rutina se efectuará con el acarreo a uno, el *flag* Z a cero y los registros BC, DE y HL sin alterar.

CAS ENTRADA DIRECTA *(CAS IN DIRECT):* BC83, 24AB

Esta llamada debe ser precedida por CAS ABRE ENTRADA. Después de eso, no debe llamarse a CAS LEE CARACTER, ya que esto impediría la ejecución de ENTRADA DIRECTA. A su entrada, HL debe apuntar al comienzo del área de datos que se van a rellenar.

Si (B802) no contiene 1 ó 3, la rutina regresará con C y Z a cero, ya que o el *buffer* no está abierto o ha sido utilizada CAS LEE CARACTER. En otro caso, (B81C/D)=HL, activando así la dirección de destino de los datos y siendo copiado el bloque en esa posición. El proceso de copia es "inteligente", siendo empleadas LDIR o LDDR de forma apropiada.

CAS RETORNO *(CAS RETURN):* BC86, 249A

Se incrementa (B81A/B), que es la cuenta de los bytes que contiene el *buffer*. Se decrementa el puntero del *buffer* en (B805/6). Esto hace que el carácter leído en último lugar esté disponible de nuevo. Sin embargo, pueden aparecer problemas si el *buffer* ha sido rellenado otra vez.

CAS EXAMINA FINAL *(CAS TEST EOF):* BC89, 2496

Se llama a CAS LEE CARACTER y la rutina retorna con C a cero y Z a uno, si se encuentra al final del fichero. En otro caso, la rutina termina a través de CAS RETORNO para hacer que el carácter esté disponible otra vez.

Todos los registros, excepto AF, son preservados.

CAS ABRE SALIDA *(CAS OUT OPEN):* BC8C, 23AB

A su entrada, B debe contener el número de bytes del nombre del fichero, HL contendrá la dirección donde se encuentra el nombre del fichero y DE debe apuntar al comienzo de un área de 2 Kbytes, que será utilizada como *buffer*.

Se inicializa el área (B847-B88B), en su mayor parte por una rutina común a CAS ABRE ENTRADA. Si ya estaba abierto algún fichero de salida, la rutina regresa con el acarreo a cero. Si la tecla ESC está pulsada, la rutina regresa entonces con C y Z a cero. Un retorno con el acarreo a uno indica que la acción prosperó y entonces HL contendrá la dirección del *buffer* de encabezamiento. El resto de los registros (incluyendo a IX) se altera.

CAS CIERRA SALIDA *(CAS OUT CLOSE):* BC8F, 2415

Si (B847)=4, se ejecutará ABANDONA SALIDA. Si (B847)=0, la rutina regresa con el acarreo a cero, indicando que no había ningún fichero de salida abierto. En otro caso, (B85D)=&FF, siendo éste el *flag* de final de fichero. Si (B85F/60)=0000, el *buffer* está vacío y entonces se ejecutará ABANDONA SALIDA. En caso contrario, se intentará escribir en la cinta y, si esto falla, la rutina retornará inmediatamente. Se llama entonces a ABANDONA SALIDA. Debe observarse que, si se pulsa la tecla ESC durante la grabación, el fichero no se cerrará.

CAS ABANDONA SALIDA *(CAS OUT ABANDON):* BC92, 242E

No se efectúa ninguna comprobación. (B847)=0. DE=(B848/9), que es la dirección del *buffer*. Si (B8CC)≠2, la rutina regresa con el acarreo a uno. En otro caso, (B8CC)=0, A=&FF y la rutina regresa con el acarreo a uno y el *flag* Z a cero.

CAS SALIDA CARACTER *(CAS OUT CHAR):* BC95, 245B

A su entrada, A debe contener un byte que se añadirá al fichero de salida.

Si (B847)≠1 o ≠2, la rutina regresará con C y Z a cero. El estado del fichero es incorrecto. En otro caso, se carga a (B847) con 2, indicando que se ha introducido un byte. Si el *buffer* está completo, su contenido se escribirá en la cinta. Si se pulsa ESC durante la grabación, provocará que la rutina retorne con C y Z a cero. El byte de salida se almacenará en la posición determinada por el puntero del *buffer* en (B84A/B); después se

incrementa el puntero. El número de bytes indicado por (B85F/60) también se incrementa.

BC, DE y HL son preservados, mientras que AF y IX se alteran.

CAS SALIDA DIRECTA *(CAS OUT DIRECT):* BC98, 24EA

Al igual que ENTRADA DIRECTA, esta rutina manipula datos en bloque. A su entrada, HL debe contener la dirección de los datos; DE debe contener la longitud de datos; BC debe contener la dirección de comienzo, si la hay, y A deberá contener el tipo de fichero.

Algún fichero debe estar previamente abierto y deberá cerrarse después de llamar a CAS SALIDA DIRECTA.

Si (B847) ≠ 1 o ≠ 3, la rutina regresa con C y Z a cero, indicando que el estado del fichero es incorrecto, ya sea porque no había ningún fichero abierto o porque se haya utilizado CAS SALIDA CARACTER. En otro caso, los datos específicos se copiarán en el *buffer* intermedio en bloques de 2 Kbytes cada uno, siendo cada bloque grabado por separado. La salida normal de rutina se produce con C a uno y Z a cero. Una salida con C y Z a cero indica un incorrecto estado del fichero, y C a cero con Z a uno indica que la tecla ESC fue pulsada.

Llamadas diversas

CAS CATALOGA *(CAS CATALOG):* BC9B, 2528

Al comienzo, DE debe apuntar hacia un área de 2 Kbytes de la RAM, que se utilizará como *buffer* temporal. Si (B802) ≠ 0, la rutina regresará, indicando que hay un fichero de entrada abierto. La dirección del *buffer* se almacena en (B803/4); se efectúa una llamada a CAS MENSAJES con A = 0 para asegurar que los mensajes se escriban en la pantalla. A continuación se lee la cinta bloque a bloque, hasta que se pulse la tecla ESC, en cuyo caso se acudirá a CAS ABANDONA ENTRADA.

CAS ESCRIBE *(CAS WRITE):* BC9E, 283F

Esta es la rutina base o primitiva utilizada por otras rutinas para grabaciones en la cinta. A su entrada, HL debe contener la dirección de los datos; DE, la longitud de datos, y A contendrá el carácter de sincro-

nismo (&16 o &2C). Observa que DE=0000 especifica que hay 6553(
bytes para escribir.

En caso de fallo o ruptura, la rutina regresará con el acarreo a cero A=0 si se pulsó la tecla ESC o A=1 para indicar otro tipo de error

CAS LEE *(CAS READ):* BCA1, 2836

Esta es la "primitiva" correspondiente a la lectura de cinta. A su entrada, HL debe contener la dirección de los datos; DE, la longitud de datos, y A debe contener el carácter de sincronismo esperado. Esta información puede venir determinada por referencia del encabezamiento, aunque estos datos son predecibles.

CAS COMPARA *(CAS CHECK):* BCA4, 2851

Esta es una función de verificación. A su entrada, HL debe contener la dirección de los datos; DE debe contener la longitud de datos, y A debe contener el carácter de sincronismo esperado. Si la comprobación tiene éxito, la rutina regresa con el acarreo a uno. Un error se indica con el acarreo a cero y con el código del error contenido en A.

Tipos de ficheros

El dato tipo de fichero es más importante de lo que pueda aparentar, ya que determina la forma en la que se manipula el fichero. El byte de tipo se construye de la siguiente manera:

Bit 0	Si es 1, el fichero está protegido
Bits 1-3	000: BASIC
	001: Binario
	010: Imagen de pantalla
	011: ASCII
Bits 4-7	000: No es ASCII
	001: ASCII

Las demás combinaciones no están definidas.

Observa que, añadiendo &24 a este byte, se forman los códigos de los caracteres que ya conoces:

BASIC	&00+&24: "$"
BASIC protegido	&01+&24=&25: "%"

Binario	&02+&24=&26: "&"
Binario protegido	&03+&24=&27: "'"

Y así sucesivamente. Sin embargo, el bit 5 de la entrada para ASCII parece ser ignorado.

Comentarios

Uno de los principales problemas en el empleo del sistema de cassette es la selección y protección de áreas disponibles temporalmente como *buffer*. Por otra parte, las llamadas pueden utilizarse con mucha sencillez, sin necesidad de preocuparse en cómo trabajan, a no ser que quieras hacer alguna travesura, como la desprotección de programas. Pero no, el método no se dará aquí, aunque no sería muy difícil conseguirlo a partir de los datos ofrecidos. Al igual que la mayoría de los sistemas de protección, éste es sensiblemente frágil.

Espacio de trabajo para el cassette

B800	*Flag* para CAS MENSAJES (0 habilita mensajes)
B801	Contador de columnas de pantalla (para los mensajes)

Bloque de control de ficheros de entrada

B802	Estado del fichero
B803/4	Dirección del *buffer*
B805/6	Puntero del *buffer*
B807/16	Nombre del fichero
B817	Número de bloques
B818	*Flag* para EOF (final del fichero)
B819	Tipo de fichero
B81A/B	Bytes en el *buffer*
B81C/D	Dirección de escritura de datos
B81E	*Flag* de primer bloque
B81F/20	Longitud de datos
B821/2	Dirección de ejecución
B823/46	Sin definir

Bloque de control de ficheros de salida

B847	Estado del fichero
B848/9	Dirección del *buffer*

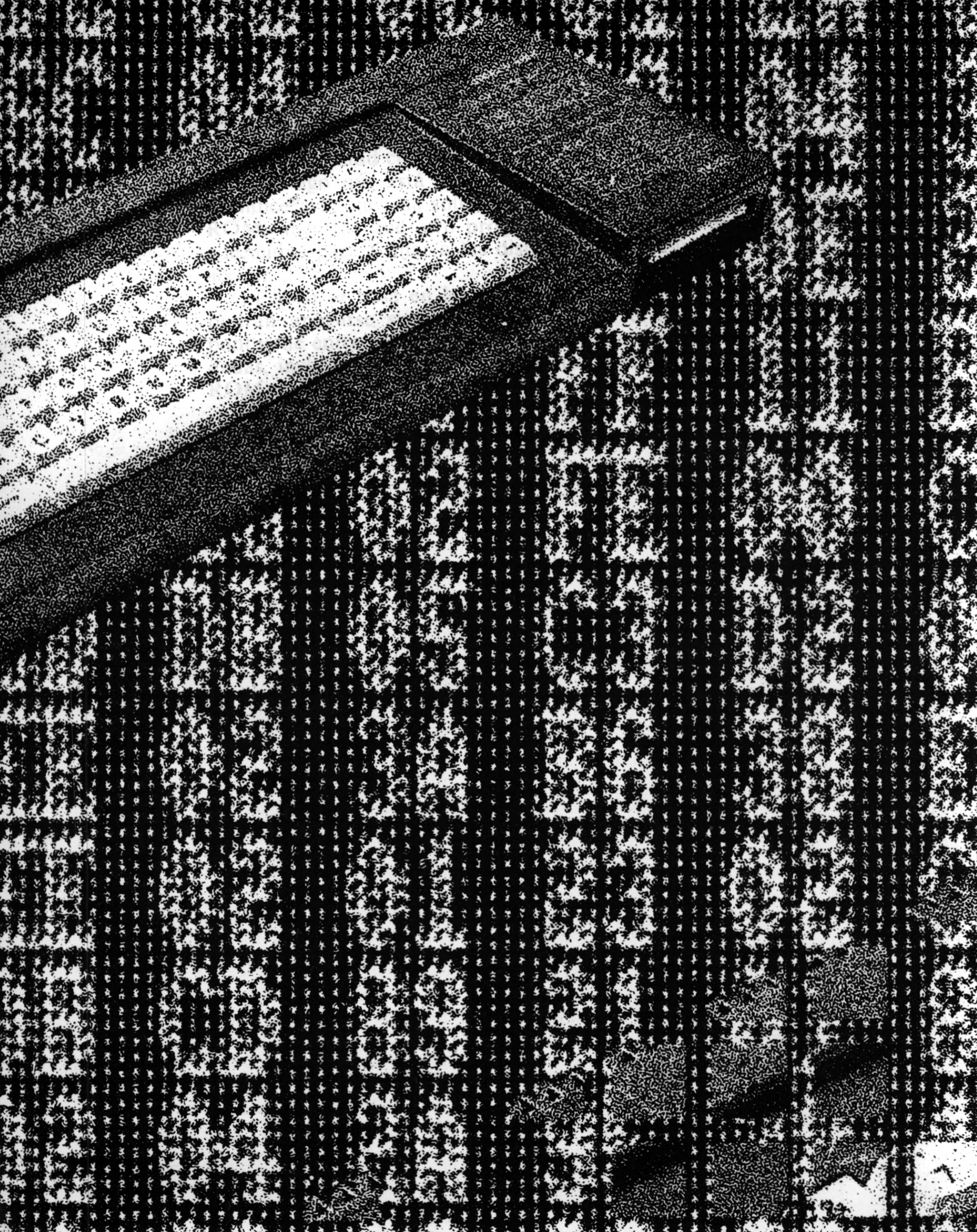
B84A/B	Puntero del *buffer*
B84C/5B	Nombre del fichero
B85C	Número de bloques
B85D	*Flag* para EOF (final de fichero)
B85E	Tipo de fichero
B85F/60	Bytes en el *buffer*
B861/2	Dirección de lectura de datos
B863	*Flag* de primer bloque
B864/5	Longitud de datos
B866/7	Dirección de ejecución
B868/8B	Sin definir

Copia del encabezamiento

B88C/9B	Nombre del fichero
B89C	Número de bloque
B89D	*Flag* de último bloque
B89E	Tipo de fichero
B89F/A0	Longitud de datos
B8A1/2	Dirección de los datos
B8A3	*Flag* de primer bloque
B8A4/5	Longitud total de datos
BA6/7	Dirección de comienzo de ejecución
B8A8/CB	Sin definir

General

B8CC	*Flag* para mensajes de ayuda
B8CD	Carácter de sincronismo
B8CE/F	Temporización
B8D0	Temporización
B8D1	Precompensación
B8D2	Velocidad
B8D3/4	Temporización



11 El monitor de sonido

El monitor de sonido ocupa la zona 1E68-2363 de la ROM inferior y utiliza el área B550-B7F9 como espacio de trabajo. Hay definidas 11 entradas para el sonido.

La secuencia de acciones necesita ser claramente entendida. Primero, un bloque de 9 bytes de datos que definen un sonido será inicializado en la RAM. Los datos serán transferidos, de forma levemente modificada, a una cola con una especie de "muesca" de aviso, al llamar a COLA SONIDO con HL apuntando al bloque mencionado. Hay cuatro de estas colas en total. La transferencia se efectuará siempre que una cola esté disponible.

Si el canal en cuestión del PSG (generador programable de sonido) está libre, la cola que aguarda entre dos "muescas" se trasladará a un *buffer* común para ese canal. Esto es efectuado por la rutina de un suceso, en el transcurso de una interrupción.

Parte del contolador de interrupciones actualiza los parámetros del área común cada centésima de segundo, transfiriendo cualquier instrucción necesaria al PSG.

Una vez que ha sido llamada COLA SONIDO, el resto del proceso es automático; es una compleja secuencia de acciones que deben ser mantenidas con precisión. Cualquier pequeña interrupción o modificación del proceso nos llevará probablemente a un caos del sistema.

Un problema específico comienza cuando se han inicializado más de cinco sonidos sucesivos para un canal determinado. El primer sonido va al

área común; los cuatro siguientes irán a las colas con muesca. Después de eso, se necesitará llamar repetidamente a EXAMINA COLA para descubrir cuándo puede hacerse una entrada más. Pero esto puede retrasar cualquier otra acción. Esperar demasiado tiempo entre comprobaciones puede provocar discontinuidad en la producción del sonido.

La pretendida solución implica el empleo de un bloque de sucesos y de una rutina asociada que provenga del mismo usuario. Una vez que se ha inicializado y activado, el suceso llamará a la rutina si hay alguna muesca libre; entonces, se podrá realizar la siguiente entrada a la cola de sonido. Este puede ser un asunto un tanto complicado, especialmente si se está utilizando más de un canal. Una vez más, la acción es enteramente automática, lo cual es conveniente para algunos casos, pero restrictivo para otros.

Sin embargo, siempre es posible actuar directamente sobre el PSG mediante MC REGISTRO SONIDO. Con esta rutina cargamos el registro definido en A con los datos contenidos en C. Esta puede ser una vía de escape, si nuestro propósito es únicamente el de experimentar.

La función de los catorce registros del PSG se puede resumir así:

R0	Período del tono, canal A, bits 0-7
R1	Período del tono, canal A, bits 8-11
R2	Período del tono, canal B, bits 0-7
R3	Período del tono, canal B, cits 8-11
R4	Período del tono, canal C, bits 0-7
R5	Período del tono, canal C, bits 8-11
R6	Período del ruido, bits 1-4
R7	Activación (0 activa, 1 inhibe)
	Bit 0: Tono canal A (0 actividad, 1 inhibición)
	Bit 1: Tono canal B
	Bit 2: Tono canal C
	Bit 3: Ruido canal A
	Bit 4: Ruido canal B
	Bit 5: Ruido canal C
	Bit 6, 7: Control de puntos de E/S. 0 para entrada. 1 para salida.
R8	Volumen canal A
R9	Volumen canal B. Se fija el nivel entre 0-&0F
R10	Volumen canal C. &1X indica la envolvente
R11	Período de envolvente, bits 1-7
R12	Período de envolvente, bits 8-15
R13	Tipo de envolvente: 0 a &0F
R14/15	Puertos de E/S

Empleado directamente, el PSG producirá una útil y variada gama de sonidos, pero el programa director debe mantener la cuenta de tiempo, ya

que no hay ninguna realimentación que compruebe si un sonido se ha completado.

Llamadas del sistema

REINICIALIZA SONIDO *(SOUND RESET):* BCA7, 1E68

El área del espacio de trabajo se pone a cero para todas las intenciones y propósitos del sistema, exceptuado los siguientes puntos:

- El bloque de suceso, el cual es del tipo asíncrono de dirección cercana y que está inicializado en el área (B555-B55B).
- La posición &1C del *buffer* de cada canal se carga con 4, indicando que hay cuatro muescas libres disponibles.
- Las posiciones 0-2 del *buffer* de cada canal se cargarán de la siguiente forma:

		A	B	C
Posición 0	Número de canal	0	1	2
Posición 1	Bit de canal	1	2	4
Posición 2	Bit de acorde	&08	&10	&20

Las envolventes no se modifican. Se silencian todos los canales.

COLA SONIDO *(SOUND QUEUE):* BCAA, 1F9F

A su entrada, HL debe apuntar a un bloque de 9 bytes que definen el sonido requerido.

Byte 0	Bit 0:	Selecciona el canal A si está a 1
	Bit 1:	Selecciona el canal B si está a 1
	Bit 2:	Selecciona el canal C si está a 1
	Bit 3:	Sincroniza con A, si está a 1
	Bit 4:	Sincroniza con B, si está a 1
	Bit 5:	Sincroniza con C, si está a 1
	Bit 6:	Retiene el sonido si es 1
	Bit 7:	Corta todos los sonidos si es 1
Byte 1		Envolvente de volumen 0-&0F
Byte 2		Envolvente de tono 0-&0F

Byte 3	Período del tono, bits 0-7
Byte 4	Período del tono, bits 8-11
Byte 5	Período del ruido, bits 0-4
Byte 6	Volumen inicial
Bytes 7, 8	Duración o contador de repetición de la envolvente.

En todos los casos, CONTINUA SONIDO es llamada para ejecutar cualquier sonido contenido en cualquier canal.

Si no se especifica ningún canal, la rutina regresará con el acarreo puesto a uno.

Si el bit 7 del byte 0 está a 1, el canal o canales especificados serán vaciados, que es el mismo efecto que conseguimos con REINICIALIZA SONIDO.

Después se comprueba si hay muescas vacías en los *buffers* del canal o canales especificados. Si todas las muescas de los canales especificados están ocupadas, la rutina regresará con el acarreo a cero. En caso contrario, los datos son transferidos a la primera muesca libre. El byte 0 se carga con los bits de canal; el byte 1 se carga con (16*Número de envolvente de volumen + Número de envolvente de tono), y los bytes 2 al 7 se cargan con los datos de los bytes 3 a 8 del bloque de datos. Los punteros de muesca se actualizan.

Cuando las muescas están disponibles, el suceso llamado —si es que existe— se ignora para introducir más datos. Además, para mantener la continuidad, dicho suceso debe atenderse a sí mismo. Si se adopta tal postura, debe asumirse que COLA SONIDO será llamada por dicho suceso.

EXAMINA COLA *(SOUND CHECK):* BCAD, 206C

A su entrada, A contendrá 1 para examinar el canal A, 2 para examinar el canal B, 4 para examinar el canal C. El estado del canal se devuelve en el mismo registro A, del siguiente modo:

Bits 0-2:	Número de muescas libres
Bit 3:	Esperando al canal A
Bit 4:	Esperando al canal B
Bit 5:	Esperando al canal C
Bit 6:	Canal retenido. Se mantiene el estado anterior
Bit 7:	Canal activo

Se ignora el suceso del usuario para sonidos.

FORMA SUCESO SONIDO *(SOUND ARM EVENT):* BCB0, 2089

A su entrada, HL debe apuntar a un bloque de sucesos de usuario, el cual debe haberse inicializado con NC INICIALIZA SUCESO. A debe contener los bits del canal relevante.

La dirección del bloque del suceso se almacenará en el área temporal del canal, pero, si hay alguna muesca libre, el byte superior de la dirección se pondrá a cero y se llamará a NC SUCESO para atender al suceso. Poniendo a cero el byte superior, se desarma el suceso, el cual se reactivará a sí mismo cuando sea ejecutado, o después de la ejecución de COLA SONIDO o de EXAMINA COLA.

PARAR SONIDO *(SOUND HOLD):* BCB6, 1ECB

Todos los canales son silenciados. Si algún canal estaba en actividad, la rutina regresará con el acarreo a uno.

CONTINUA SONIDO *(SOUND CONTINUE):* BCB9, 1EE6

Los canales activos que estaban retenidos serán liberados si el bit del canal estaba a 1 en el registro A.

EJECUTA SONIDO *(SOUND RELEASE):* BCB3, 204A

A su entrada, A debe contener los bits de canal para los canales que van a liberar. Se llama a CONTINUA SONIDO y se actualizan los *flags* y punteros.

ENVOLVENTE VOLUMEN *(SOUND AMPL ENVELOPE):* BCBC, 2338

Al comienzo, A debe contener un número de envolvente y HL debe apuntar a un bloque de datos de no más de 16 bytes, como el siguiente:

Byte 0:	Número de secciones
Bytes 1-3:	Sección 1
Bytes 4-6:	Sección 2
Bytes 7-9:	Sección 3
Bytes 10-12:	Sección 4
Bytes 13-15:	Sección 5

Si el byte 0=0, la envolvente constará de un nivel de sonido constante que se mantendrá durante dos segundos.

Se pueden especificar envolventes de tipo *software* o de tipo *hardware*. Para una "envolvente *software*", los bytes de una sección son:

Primer byte:	Cuenta de pasos (1-127)
Segundo byte:	Altura del paso
Tercer byte:	Tiempo del paso

Si la cuenta de pasos es cero, el nivel de volumen vendrá dado por altura de paso.

Para una "envolvente *hardware*":

Primer byte:	Forma de envolvente +&80
Segunto byte:	Período de la envolvente (bajo)
Tercer byte:	Período de la envolvente (alto)

Los datos se relacionan directamente con los registros 11-13 del PSG.

Suponiendo válido el número de envolvente contenido en A, los datos de la envolvente se copian sin alteración alguna del área de almacenamiento de la envolvente. La rutina regresará con el acarreo a uno y con HL conteniendo la dirección del siguiente área de almacenamiento de envolvente. El acarreo a cero indicará algún fallo en los datos.

ENVOLVENTE TONO *(SOUND TONE ENVELOPE)*: BCBF, 233D

Esta rutina es muy similar a la anterior, excepto en el significado de los datos. Existen dos formatos para las secciones:

Byte 1:	Cuenta de pasos (0-239)
Byte 2:	Altura de paso (−127 a 127)
Byte 3:	Tiempo del paso (en centésimas de segundo)

La altura del paso se añade al nivel actual del volumen. Alternativamente:

Byte 1: 240 a 255
Byte 2: −127 a 127
Byte 3: Tiempo del paso

El período del tono se calcula así: 256*(byte 1-240)+byte 2.
Si el bit 7 del primer byte de los datos de envolvente es un 1, la envolvente se repetirá.

DIRECCION ENVOLVENTE V *(SOUND A ADDRESS):* BCC2, 2349

Esta rutina convierte el número de envolvente de volumen contenido en la entrada de A, en la dirección de los datos de la envolvente que estará guardada en HL a la salida. Una salida con el acarreo a cero significa que el número contenido en A no será válido.

DIRECCION ENVOLVENTE T *(SOUND T ADDRESS):* BCC5, 234E

Todo igual que en la rutina anterior, pero refiriéndose a las envolventes de tono.

Comentarios

Las rutinas de sonido son muy ingeniosas, quizá demasiado ingeniosas dentro de su bondad. La apreciable automatización de sus elementos puede ser frustrante para aquel que desee experimentar, a menos que se eviten completamente estas rutinas.

Este comentario puede ser por una larga experiencia con sistemas de sonido relacionados casi enteramente con el *software*, incluyendo también a los dispositivos PSG. La escritura de las rutinas y datos para esos sistemas puede ser laboriosa, pero el programador puede sentirse a cargo de la situación, mientras que con acciones automatizadas tiene mucho menos control.

A pesar de todo, el sistema CPC es versátil —más versátil de lo que le parecerá a un usuario de BASIC— y puede producir sonidos realmente interesantes.

Podría apreciarse que la generación de música requiere menos datos de lo que se piensa. Una vez que se ha establecido la forma de una envolvente, las únicas variables importantes son el tono y la duración. El nivel de volumen puede diferir entre los distintos canales, para obtener un pequeño efecto de balance, pero puede dejarse habitualmente a un nivel constante. Lo que se necesita en estas circunstancias es una rutina que examine las variables esenciales e inserte su contenido en los bloques de datos utilizados para emitir las notas.

Espacio de trabajo del monitor de sonido

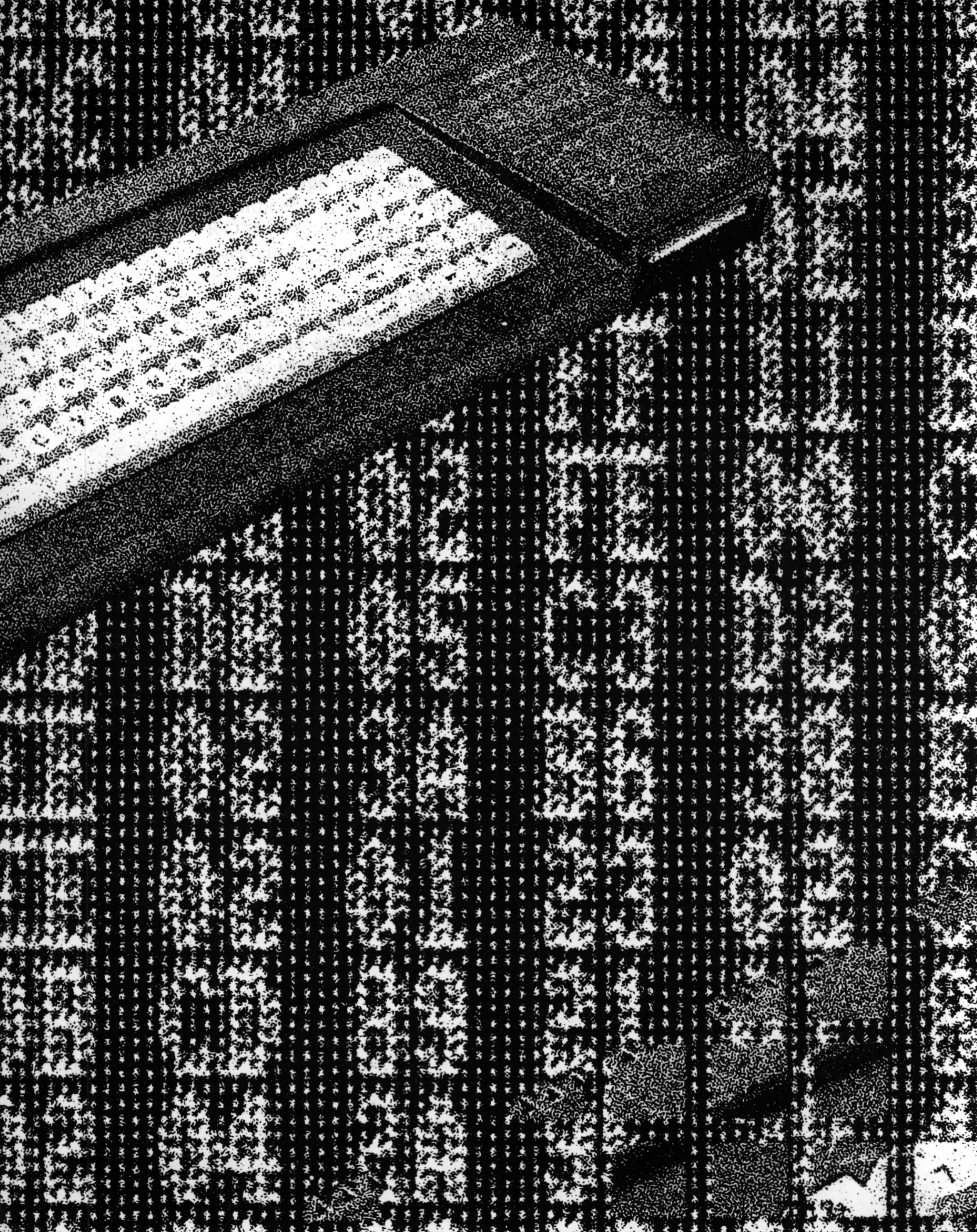
B550	*Flags* para nuevas entradas
B551	*Flags* para la retención de canales activos
B552	*Flags* para los canales activos
B553	Contador del control de interrupciones
B554	Contador de acciones pendientes
B555-B55B	Bloque de sucesos
B55C-B59A	*Buffer* del canal A
B59B-B5D9	*Buffer* del canal B
B5DA-B618	*Buffer* del canal C
B619	Byte de activación del PSG
B61A-B709	Envolventes de volumen
B70A-B7F9	Envolventes de tono

Formato del *buffer* de un canal

&00	Número de canal
&01	Bit del canal
&02	Bit de acordes del canal
&03	*Flags* de estado del canal
&04	Llamadas para el establecimiento del tono si bit 0=1
&05	Contador 1 de interrupciones
&06	Contador 2 de interrupciones
&07	Llamadas para el establecimiento del volumen si bit 0=1
&08-&09	Duración negada (complementada)
&0A-&0B	Dirección de la envolvente de volumen
&0C	Número de secciones de la envolvente de volumen
&0D-&0E	Dirección de la sección de la envolvente de volumen
&0F	Volumen

&10	Tipo de envolvente (envolvente *hardware*)
&11-&12	Dirección de la envolvente de tono
&13	Número de secciones de la envolvente de tono
&14-&15	Dirección de la sección de la envolvente de tono
&16-&17	Período del tono
&18	Cuenta de pasos
&19	Número de muesca actual
&1A	Cuenta
&1B	Muesca en uso
&1C	Muescas libres
&1D-&1E	Dirección del bloque de sucesos
&1F-&26	Muesca 1
&27-&2E	Muesca 2
&2F-&36	Muesca 3
&37-&3E	Muesca 4

Las direcciones que aparecen a la izquierda de la lista son relativas al *buffer* del canal.



12 Las ROM externas

De cuando en cuando ha sido necesario mencionar las ROM externas, pero por ahora debemos analizarlas desde una perspectiva más amplia. Sólo los más dedicados entusiastas querrán aventurarse en este tema, en el que se combina el *hardware* y el *software* de forma experta. Aun así, el sistema merece un examen a fondo.

Nominalmente se pueden añadir al sistema hasta 252 ROM superiores externas, para proveerle de programas adicionales, estando el límite determinado por el formato de LLAMADA y de LLAMADA HL, que interpretan de forma especial los números de ROM &FC-&FD. En la práctica, el número de ROM añadido tiende a ser mucho menor.

A cada ROM se le debe dar un número y, cuando ese número salga por DFXX, la ROM deberá activarse. Esto no significa que la memoria ofrecerá inmediatamente datos al *bus*. Esto sólo lo puede hacer cuando esté activada, la señal ROMEM sea cierta (activa a nivel bajo) y el bit A15 de dirección esté a uno. En estas circunstancias, la línea ROMDIS debe ponerse a nivel alto antes de ser activada la ROM externa, asegurando así que la ROM interna (inferior y superior) entra en un estado de alta impedancia antes de que la ROM externa trate de ocupar el *bus* de datos. No atender a este detalle puede provocar daños físicos en los dispositivos.

La ROM interna superior se desactiva así cuando una ROM externa está activada, incluso si a la ROM externa se le ha asignado el número 0, el cual se asocia corrientemente a la ROM interna superior. De hecho, la

ROM interna superior se activa cuando se produce una salida con DFXX, lo cual no significa que se esté señalando a una ROM externa.

A las ROM primarias se les puede asignar cualquier número y es posible dar cabida a un programa primario de hasta 64K, de forma homogénea, sin más que utilizar cuatro ROM de 16K, numeradas consecutivamente. LLAMADA LATERAL permite un sencillo acceso entre las ROM de dicho grupo.

A las ROM secundarias se les debe dar números en el margen 1-7, de forma que puedan ser inicializadas por NC ROM SECUNDARIA, como se describirá más adelante.

La distinción entre ROM secundaria y primaria puede aclararse ahora un poco más. Sólo se puede activar una ROM primaria cada vez, mientras posee el control, y sólo se permite la existencia de un único agente controlador en un momento dado. Por otra parte, las ROM secundarias pueden ser llamadas temporalmente como soporte de una rutina primaria.

Entrando en C006, una ROM primaria debe modificar los datos de la dirección de almacenamiento transferidos en BC, DE y HL para reclamar una adecuada cantidad de espacio de trabajo. El intérprete de BASIC, por ejemplo, reclama dos áreas iniciales. Un área baja en 0040-016F, y un área alta en AC00-B0FF. Con esto se marcan los límites del área principal de trabajo, la cual debe contener los programas BASIC, variables, cadenas y los *buffers* para el cassette. Esta área es denominada depósito de memoria y su distribución debe hacerse con cuidado.

Habiendo reclamado su propio espacio de trabajo, el programa primario debe invitar, a cualquier ROM secundaria que pretenda utilizar, a que reclame un espacio de trabajo para sus propias necesidades. NC INIC SECUNDARIA hará esto para una ROM en concreto, mientras que NC ROM SECUNDARIA inicializará espacio de trabajo para todas las ROM secundarias que haya en el sistema.

MC CARGA Y EJECUTA permite que un programa sea cargado en la RAM y que sea tratado como un programa primario. Pero también existe una provisión para programas en RAM que actuarán en el papel de secundarios. Dichos programas son conocidos como "Extensiones del Sistema Residente" o también RSX *(Resident System Extensions)*, y están capacitados para reservar su propia área de datos.

Ordenes

Aunque cualquier posición de un programa puede ser accesible mediante saltos o llamadas, a veces es más conveniente utilizar ciertas órdenes. La ROM interna superior tiene, de hecho, una única orden BASIC. Esta accede a la entrada de inicialización en C006. En cambio, otras ROM pueden definir una multitud de órdenes especiales. Cada una de dichas

órdenes requiere estar precedida de "|" (SHIFT+@), para distinguirse de las órdenes del BASIC.

Las primeras posiciones de una ROM externa deben obedecer al siguiente formato:

C000	Tipo de ROM
C001	Número-marca de la ROM
C002	Número de la versión
C003	Nivel de modificación de la ROM
C004/5	Dirección de la tabla de órdenes externas
C006	Bloque de saltos para las órdenes

Los tipos de ROM son:

0:	Primaria
1:	Secundaria
2:	Para una extensión de la ROM
&80:	Es la ROM propia del sistema

El formato de la ROM necesita ilustrarse con un ejemplo. Supón que una ROM nos provee de un controlador intensificado para impresora, y que puede generar secuencias de código que inicializan la impresora en un modo particular de trabajo. En términos simplistas, el comienzo de dicha ROM puede parecerse a esto:

C000:	01	01	01	01	
C004:	00	C1			La tabla de órdenes comienza en C100
C006	C3	00	C2		Inicialización en C200
C009:	C3	00	C3		DOBLE pasada en C300
C00C:	C3	00	C4		ENFASIS en C400
C00F:	...	...	...		Otras conexiones

Tabla de órdenes:

C100	49	4E	49	C3			INIC
C104	44	4F	42	4C	C5		DOBLE
C109	45	4E	46	41	53	D3	ENFASIS
C110	...	...	...				Otros nombres de órdenes

Las palabras de las órdenes deben consistir en caracteres alfabéticos en mayúsculas (aunque los puntos también parecen adaptarse), y debe añadirse &80 al código de la última letra de cada palabra. La tabla de órdenes deberá terminarse con un cero.

Para la inicialización, la ROM se comienza a leer a partir de C006, no estando implicada ninguna orden. Sin embargo, si la palabra ENFASIS

había sido impuesta por el usuario y si se llamó a NC BUSCA ORDEN, la rutina regresará con la dirección C00C de la entrada en HL y el número apropiado de ROM en C. Después, SALTO HL accederá a la rutina requerida.

Lo mejor es hacer sencillas las palabras de las órdenes; aun así, se admiten hasta dieciséis caracteres significativos.

Existen similares facilidades disponibles para los programas en la RAM, pero en este caso no hay obligaciones para la localización de la tabla de órdenes. La conexión con la tabla se hace a través de un blique de referencia de cuatro bytes, que en el caso de las ROM secundarias se sitúa inmediatamente debajo del área dada como espacio de trabajo de la ROM.

Con estas amplias facilidades para la extensión de programas, es natural que haya habido dudas respecto a la posibilidad de extender la RAM de manera análoga. Esto, desafortunadamente, no es fácil de conseguir. Cuando se ejecuta una operación de escritura, la RAM queda seleccionada y también se activa el *buffer* de salida de la RAM, pero esta activación no permite acceder al conector de expansión y no hay forma de anular esto último. Se podría conseguir una escritura simultánea en una RAM interna y otra externa si pudiéramos decodificar las señales disponibles para alcanzar la activación necesaria, pero, en todo caso, necesitamos alguna línea de selección de la memoria RAM externa, tal como la que puede ofrecer un *latch* activado convenientemente.

Claro está que es posible inicializar una RAM externa y leerla después como si se tratara de una ROM, pero éste es ya otro asunto bien diferente. Dicho procedimiento puede ser útil mientras se esté desarrollando o diseñando alguna ROM, o en conexión con otro sistema de ordenador, pero por aquí empezamos a movernos en aguas oscuras.

Rutinas

Las cuatro rutinas del núcleo relevantes para el sistema de órdenes son:

NC INTRODUCE RSX *(KL LOG EXT):* BCD1, 02A1

A su entrada, HL debe apuntar a un bloque de cuatro bytes de la RAM que no es utilizado en otro caso. Para una ROM secundaria, B=0 y C contiene el número de la ROM, pero, para una RSX, BC debe contener la dirección de la tabla de órdenes.

PUSH HL
DE=(B1A6/7)

(B1A6/7)=HL
(HL)=E
(HL+1)=D
(HL+2)=C
(HL+3)=B

Esto inicia el área de cuatro bytes. Los primeros dos bytes apuntan a la última de dichas áreas, de forma que por simple exploración es posible comenzar en B1A6/7 y rastrear por turno todas las demás áreas.

Un procedimiento ligeramente diferente es utilizar para las ROM secundarias y las RSX la última aplicación, es decir, si B≠ 0.

NC INIC SECUNDARIA *(KL INIT BACK):* BCCE, 0332

A su entrada, C debe contener el número de la ROM que hay que inicializar; DE debe contener la dirección del byte libre más bajo en la memoria, y HL debe contener la dirección del byte libre más alto de la memoria. La información de las direcciones se transfiere a la ROM primaria por medio de la rutina en 0077, la cual llama a la ROM, y la ROM primaria debe transferir entonces las direcciones a NC INIC SECUNDARIA, modificándolas quizá para reclamar su propio espacio de trabajo.

Si el contenido de C está fuera del margen 1-7, que son los números permitidos para las ROM secundarias, la rutina abandonará y regresará inmediatamente. En otro caso, se llama a SELECCIONA ROM con A=C. Si (C000) AND 3≠1, la ROM no es del tipo secundario, y la rutina regresa vía ROM ANTERIOR.

En caso contrario, se guarda BC en la pila, y se llama a la ROM en C006, que es el punto de entrada para la inicialización. La ROM debe modificar los punteros de dirección para reclamar el espacio de trabajo que requiera.

Los valores modificados se devuelven en DE (LOMEM) y en HL (HIMEM), valor inferior y superior, respectivamente, de la memoria utilizable. Se guarda DE en la pila y DE=HL+1. Entonces se carga HL con B1AA+2*(B1A8), siendo (B1A8) el número de la ROM actualmente seleccionada. DE se transfiere a la posición así definida. Este será el nuevo valor de HIMEM, señalando el comienzo del espacio de trabajo reservado.

A continuación se llama a NC INTRODUCE RSX con HL=DE−4, B=0 y C conteniendo el número de ROM. Esto inicializa el bloque de cuatro bytes en el área inmediatamente por debajo del espacio de trabajo.

HL se actualiza de forma que apunte a la posición debajo del bloque de cuatro bytes, y DE y BC se recuperan de la pila. La rutina regresa entonces vía ROM ANTERIOR.

Observa que la tabla se inicia en B1AA+2*(B1A8), es decir, el fondo de cada área de datos, sin incluir el bloque de cuatro bytes. La cima o tope del área de datos no se define de la misma manera, pero puede observarse por el proceso de inicialización de la ROM.

Advierte también que no es cierto que a una ROM secundaria le corresponda siempre el mismo área de datos, aunque algunas rutinas de acceso transfieren la dirección de comienzo del espacio de trabajo en IX cuando se llama a la ROM (véase “Rutinas de la RAM”).

NC ROM SECUNDARIA *(KL ROM WALK):* BCCB, 0329

Esta rutina llama a NC INIC SECUNDARIA con sucesivos valores de C decreciendo desde 7 hasta 1, inicializando con ello todas las ROM secundarias disponibles. Las condiciones de entrada son las mismas que para NC INIC SECUNDARIA, excepto que no se necesita dar ningún valor a C.

NC BUSCA ORDEN *(KL FIND COMMAND):* BCD4, 02B2

Se accede a ella con HL apuntando a la palabra de una orden inicializada en RAM. La orden puede estar en la ROM, ya que la primera acción es copiar los dieciséis bytes de la palabra en (B196-B1A5). El bit 7 del último byte a copiar estará puesto a uno, después HL=(B1A6), A=L. La rutina salta a 02D5 y después a 02C5 si (HL)≠0.

02C5: Se guarda HL en la pila y BC=(HL+2, HL+3). Esta es la dirección de la tabla de órdenes para una RSX o el número de una ROM secundaria. Se llama 02F4 para procesar este dato coherentemente.
Si 02F4 regresa con el acarreo a uno, la palabra-orden se ha emparejado y la rutina regresa con DE conteniendo el enlace de la cadena que se guardó en la pila con HL (esto es, para vaciar la pila), mientras que HL contiene la dirección de entrada y C el número de ROM.
En otro caso, el enlace de cadena se transfiere a HL y HL=(HL) toma el nuevo enlace de la cadena, de forma que se pueda comprobar la tabla de la siguiente palabra-orden.

02D5: Si HL≠0, la rutina inicia un bucle con 02C5 para una nueva búsqueda. Si HL=0000, es que se ha alcanzado el final de la

cadena y necesitan comprobarse otras posibilidades. C=&FF y comienza un nuevo bucle.

02DA: C=C+1. Se llama a NC TIPO ROM para determinar el tipo de ROM. Si este es 0 o &80, la primacía del sistema, se llamará a 02F4. Si de allí se regresa con el acarreo a uno, se llama a MC EJECUTA PROGRAMA. Esto último quiere decir que puede utilizarse una orden RSX para seleccionar una nueva ROM primacía. En caso de no ser de clase &80, o si C=0, entonces la rutina inicia un bucle con 02DA para intentarlo con otra ROM. Excepto en este caso, la rutina regresa con el acarreo a cero.

02F4: HL=C004, apuntando a la dirección de la tabla de una orden en una ROM, pero, si B≠0, se accederá a una tabla RSX y entonces HL=BC, C=&FF.
Se llama a NC SELECCIONA ROM, se guarda BC en la pila, DE=(HL), HL=HL+2. Se intercambian DE y HL, y la rutina salta a 0321. En este punto, DE contiene la dirección de enlace de la tabla de la orden y HL contiene la dirección de la tabla.

030A: BC=B196, que es el comienzo de la copia de la palabra-orden.

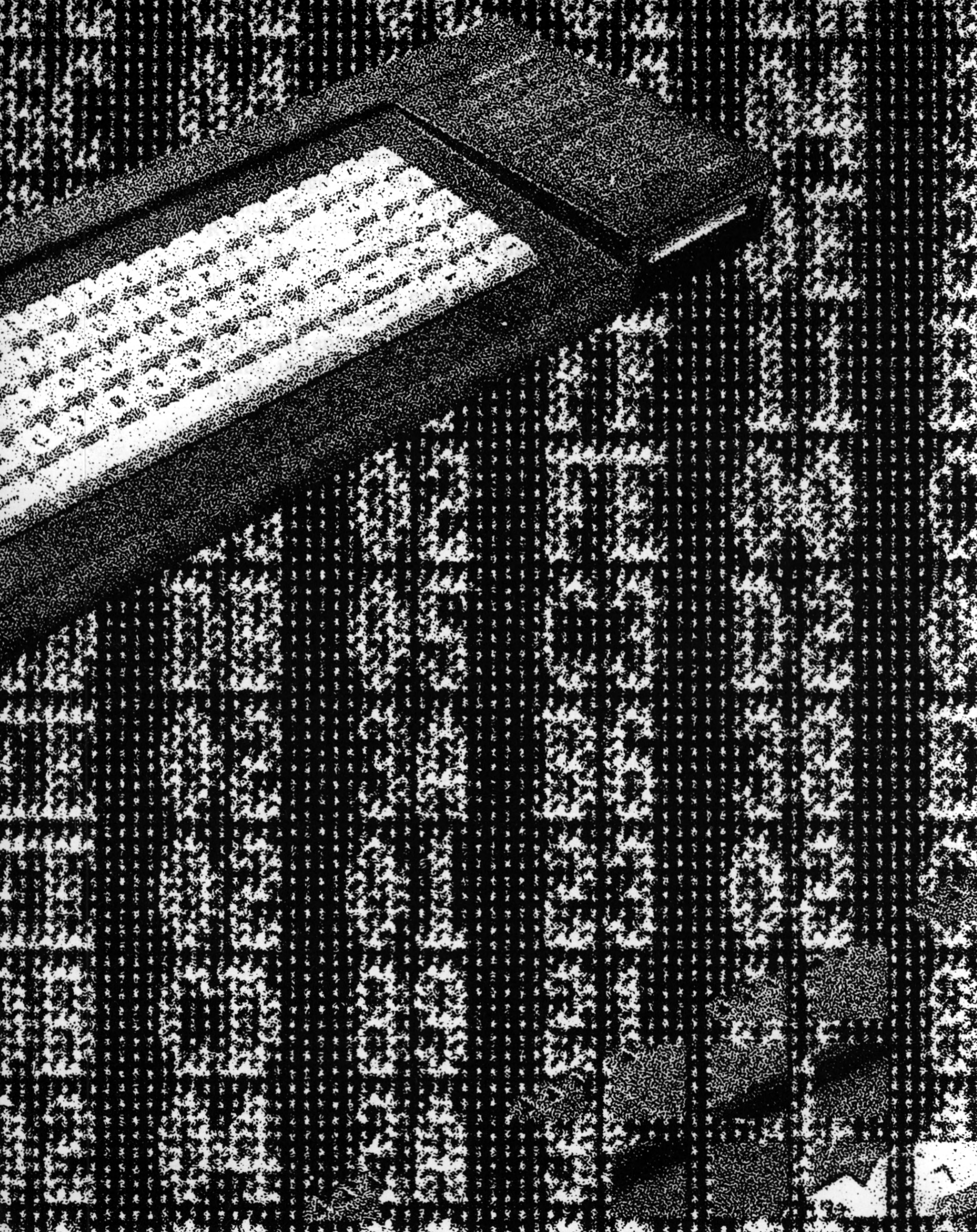
030D: A=(BC). Si A≠(HL), la rutina salta a 0319. No ha habido emparejamiento. En otro caso, HL y BC son incrementados y la rutina sigue un bucle con 030D hasta que el bit 7 de A se encuentre a 1, indicando el final de la palabra. Si se alcanza esta última condición, querrá decir que se ha encontrado la pareja de la palabra-orden requerida. Se intercambian DE y HL, y la rutina salta a 0325 con el acarreo a uno (desde el bit 7 de A). HL apunta a la entrada del bloque de saltos y C contiene el número de ROM.

0319: A=(HL), HL=HL+1. La rutina inicia un bucle con 0319 hasta que el bit 7 de A sea uno, señalando el final de una palabra. Al finalizar el bucle, DE=DE+3, avanzando así hasta la siguiente entrada del bloque de saltos.

0321: Si HL≠0000, entonces se regresa a 030A para probar con la siguiente palabra.

0325: Se recupera BC de la pila y se sale vía ROM ANTERIOR.

Esto completa las rutinas relevantes de las funciones para palabras-orden.



13 Soporte del BASIC

Las entradas a la sección 2A98-37FF de la ROM inferior no están definidas oficialmente, porque en realidad son más una parte del intérprete de BASIC que un sistema operativo. De todos modos, las 49 entradas contienen muchísimos tesoros, especialmente para aquellos cuyos programas impliquen el uso de matemáticas.

La primera entrada, vía BD3A, accede al sistema de EDICION, que es demasiado especializado como para resultar de interés general y que se emplea sobre todo para la interpretación de varias combinaciones de teclas y para la modificación de datos contenidos en un *buffer*, cuyo comienzo se define en HL a la entrada.

Las entradas relativas a aritmética de coma flotante son de importancia mucho más general.

La coma flotante

Se utiliza un sistema de 5 bytes de coma flotante. Por ejemplo:

86 65 2E E0 D3

El primer byte es el exponente. Sustrayendo &80, obtenemos 6, de

forma que el valor del exponente es 2^6=64. Los bytes restantes constituyen la mantisa, y el valor total del número se halla multiplicando el valor del exponente por la mantisa.

El bit más significativo del segundo byte es el bit de signo. En este caso es 0, así que el número es positivo. Sin embargo, el verdadero valor del bit en términos numéricos es 1 siempre, de modo que el valor de los últimos cuatro bytes —la mantisa— es:

E62EE0D3 = 3,845,054,675

La mantisa, sin embargo, se expresa en "binario fraccional", lo que significa que el bit más significativo tiene un valor de 1/2; el siguiente bit, un valor de 1/4, y así sucesivamente. Para hallar el valor real de la mantisa hay que dividir entre 2^32. Multiplicando por 64 el resultado, se obtiene el valor del número de la coma flotante, igual a 57.2957795 = 180/PI.

Un valor igual a cero constituye un caso especial, al ser el exponente 0 y la mantisa irrelevante. Esto puede considerarse como el siguiente paso desde un exponente igual a &01, que tiene un valor de 2^ −127. Combinado con el valor mínimo de la mantisa, que es 0.5, puede representarse un valor total de 2^ −128. El máximo valor posible que puede mostrarse es un pelo menor que 2^127.

Los números en coma flotante se almacenan con los bytes en orden inverso, con el byte menos significativo de la mantisa en primer lugar y, por último, el exponente. Puede hallarse un número de ejemplos en la ROM, en el área 2E18-37FF. Algunos están colocados en medio de una sección de código, que hace difícil el desensamblado. Esto es porque la rutina "series exponenciales" toma constantes de las posiciones siguiendo la instrucción de llamada. Por ejemplo:

```
CD   A9   32                  CALL 32A9
04                            4 entradas en la tabla
4C   4B   57   5E   7F        0.4342597
0D   08   9B   13   80        0.5765815
23   93   33   76   80        0.9618007
20   3B   AA   38   82        2.8853901
```

La rutina continúa en la posición siguiendo la tabla, que en este caso se toma de la rutina LOG.

Cuando una mantisa se lleva a los registros, ocupa normalmente DEHLC, utilizándose C para recoger cualquier bit de acarreo en operaciones de desplazamiento a la derecha. Pueden necesitarse para propósitos indirectos.

Hay tres posiciones definidas en la RAM para contener números en coma flotante:

CONTENIDO 1: B8E5-B8EC
CONTENIDO 2: B8ED-B8F1
CONTENIDO 3: B8F2-B8F6

Sirven primordialmente para la utilización del intérprete de BASIC.

Se puede acceder a la aritmética de enteros y a la de coma flotante sin grandes preocupaciones sobre los detalles de trabajo, pero se han añadido algunos comentarios a la siguiente tabulación para ayudar a aquellos que queráis examinar las rutinas más profundamente. En el apéndice encontraréis un programa en BASIC que os ayudará en tales investigaciones.

Localización de las entradas

Las entradas al bloque de saltos apropiadas para esta área utilizan el código (&EF) del SALTO ROM INFERIOR en vez del código (&CF) de INF SALTO, que se emplea para el resto del bloque de saltos, pero se aplican las mismas reglas: al saltar a una entrada debe haber una dirección de regreso almacenada en la pila para permitir la continuación, una vez que se haya ejecutado la rutina llamada.

Se utilizará una especial nomenclatura para datos de coma flotante, significando CF(X) que un número en coma flotante es apuntado por el contenido del registro de índice X. Las direcciones actuales de la rutina no se proporcionan. Pueden hallarse fácilmente examinando las entradas del bloque de saltos a las direcciones establecidas.

BD3D DE y HL a la entrada apuntan hacia dos números en coma flotante (o a las áreas donde es probable que existan los números en coma flotante). CF(DE) se copia en CF(HL). A la salida, A contiene el exponente del número copiado. El acarreo se pone a uno (observa que CF(HL) debe estar en la RAM).

BD40 A su entrada, DE apunta al área RAM de números de coma flotante y HL contiene un número binario sin signo en el rango 0-65535. CF(DE) se convierte en coma flotante equivalente al número contenido en HL. A su salida, HL=DE en la entrada; DE se altera, y A contiene el byte más significativo de la nueva mantisa.

BD43 A la entrada, HL apunta a un número binario de cuatro bytes en la RAM. El número, tratado como un entero, se sobreescribe por su equivalente en CF. A la salida, HL apunta al nuevo número y A contiene el byte más significativo de su mantisa.

BD46 Esta llamada se utiliza por la orden CINT del BASIC. A la entrada, HL apunta a un número CF en la RAM, de valor comprendido en el intervalo +/− 32767. La parte entera del número se carga en HL como un número redondeado al entero más cercano en complemento a 2. A la salida, A contiene el byte

de signo del número CF. El acarreo se pone a uno, a menos que se produzca una sobrecarga por ser un número demasiado grande.

BD49 A la entrada, HL apunta a un número CF en la RAM. BD4C es llamada para convertir el número a la forma entera. Si el resultado tiene un resto mayor que 0,5, o bien si el número CF es negativo, el entero se incrementa. A la salida, C contiene el número de bytes distintos de cero del entero.

BD4C Esta llamada se utiliza por la orden FIX del BASIC. A la entrada, HL apunta hacia un número CF en la RAM. El número se trunca a la forma entera señalada, sobreescribiéndose el resultado a la mantisa del número original. A la salida, C contiene el número de bytes distintos de cero en el entero. A contiene &FF para un número negativo y 0 para un número positivo.

BD4F Esta llamada se utiliza por la orden INT del BASIC. Es prácticamente idéntica a BD49, con la excepción de que sólo es sensible al signo, y el resto se ignora.

Las llamadas anteriores requieren alguna explicación, pero la siguiente es diferente. Se utiliza para preparar salidas en decimal, aunque no realiza el proceso de salida propiamente dicho.

Se utiliza un algoritmo interesante para calcular el número de cifras decimales en la parte entera del número procesado. El valor real del exponente se halla sustrayendo &80 y el resultado se multiplica por 77/256, que es una aproximación estricta al logaritmo decimal de 2. La parte entera del resultado establece el número de cifras decimales requeridas.

El cálculo puede escribirse:

$$\log_{10} N=(\log_2 N)*(\log_{10} 2)$$

donde N es el número en cuestión.

Se resta 9 del número de cifras decimales, ya que sólo pueden ofrecerse 9 decimales. Si el resultado es distinto de cero, el número se multiplica o se divide por potencias de 10, hasta que quede comprendido entre 312500 y 10^9. (312500-(10^7)/32).

BD52 A la entrada, HL apunta hacia un número en CF en la RAM. El número se procesa como se ha descrito anteriormente y se coloca en lugar de CF(HL). HL se ajusta entonces para apuntar el byte más significativo.

Esta es, de todas las rutinas de soporte del BASIC, la más difícil de utilizar, por lo que es preferible una aproximación alternativa.

BD55 A la entrada, A contiene un valor índice y HL apunta a un número CF en la RAM. El número se multiplica por 10^A.

A puede estar comprendido entre −127 y 127, pero los valores que excedan del intervalo +/−76 no tendrán sentido. El resultado sustituye a CF(HL). A la salida, BC y DE son alterados y A contiene el byte del signo de la mantisa resultante.

BD58 A la entrada, DE y HL apuntan hacia números CF, los últimos de la RAM. Se realiza la operación CF(HL)=CF(HL)+CF(DE). A la salida, BC y DE están alterados y A contiene el byte del signo de la mantisa resultante.

BD5B Como BD58, pero CF(HL)=CF(HL)−CF(DE).

BD5E Como BD58, pero CF(HL)=CF(DE)=CF(HL).

BD61 Como BD58, pero CF(HL)=CF(HL)*CF(DE).

BD64 Como BD58, pero CF(HL)=CF(HL)/CF(DE).

BD67 A la entrada, A contiene un índice y HL apunta a un número CF en la RAM. A se añade al exponente del número, multiplicando efectivamente el número por 2^A. A puede tomar cualquier valor comprendido entre −127 y +127. A la salida, A contiene el nuevo exponente.

BD6A A la entrada, DE y HL apuntan hacia dos números CF.

- Si CF (DE)=CF(HL), la rutina sale con el acarreo a cero y el *flag* Z a uno. A=0.
- Si CF(DE)<CF(HL), la rutina sale con el acarreo a cero y el *flag* Z a cero. A=1.
- Si CF(DE)>CF(HL), la rutina sale con el acarreo a uno y el *flag* Z a cero. A=&FF.

Los números CF no se modifican.

BD6D Se niega CF(HL).

BD70 Si CF (HL)=0, regresa con A=0.
Si CF(HL)>0, regresa con A=1.
Si CF(HL)<0, regresa con A=&FF.

BD73 Introducida con A=0, carga la condición RAD (radianes). Si se introduce con A=1, carga la condición DEG (grados).

BD76 CF(HL)=PI.

BD79 CF(HL) es sustituido por su raíz cuadrada (SQR).

BD7C CF(HL)=CF(HL)^CF(DE).

BD7F CF(HL) se sustituye por su logaritmo natural. HL es preservado..

BD82 Como BD7F, pero el logaritmo está en base 10.

BD85 CF(HL)=e^(CF(HL)); (EXP).

BD88 CF(HL)=SIN (CF(HL)).

BD8B CF(HL)=COS (CF(HL)).

BD8E CF(HL)=TAN(CF(HL)).

BD91 CF(HL)=ATN (CF(HL)).

Las limitaciones y reglas para las versiones en BASIC son aplicables a estas funciones en general.

BD94 Esta es similar a BD43, pero trabaja sobre enteros de 5 bytes. El

byte menos significativo del entero se descarta, puesto que queda fuera del límite de resolución del sistema.

BD97 El número aleatorio se fija como 076C6589.

BD9A El número aleatorio se fija como antes y después se efectúa una XOR con CF(HL).

BD9D El número aleatorio se actualiza y se copia en CF(HL).

BDA0 El número aleatorio se coloca desde CF(HL).

BDA3 B=H. Si H<0, se niega. C=2, A=0.

BDA6 BC=0002, E=0.

Las dos últimas llamadas sólo son significativas para el intérprete de BASIC.

BDA9 Si H<0, es negado y la rutina regresa. En otro caso, si H>0, HL es negado.

Ahora llegamos a las operaciones enteras.

BDAC HL=HL+DE.

BDAF HL=HL−DE.

BDB2 HL=DE−HL.

BDB5 HL=HL*DE. Se utilizan valores absolutos. Si los signos de HL y de DE son distintos, B se hace negativo. Le sigue BDA9.

BDB8 HL=HL/DE, el resto en DE.

BDBB DE=HL/DE, el resto en HL.

BDBE HL=HL*DE.

BDC1 HL=HL/DE, el resto en DE. Se utilizan valores absolutos.

BDC4 Si HL=DE, regresa con A=0.
Si HL>DE, regresa con A=1.
Si HL<DE, regresa con A=&FF.

BDC7 HL es negado.

BDCA Si HL=0, regresa con A=0
Si HL>0, regresa con A=1.
Si HL<0, regresa con A=&FF.

Manejando llamadas para matemáticas

No os decepcionará la matriz de llamadas para matemáticas. Aprovechándolas al máximo, podréis obtener muchos códigos recurrentes. No hay que pensar que las descripciones ofrecidas anteriormente cubren todas las posibilidades. Utiliza el programa del apéndice para explorar los detalles. (Las descripciones se basan en el examen de las rutinas y es fácil echar en falta puntos aquí y allí...)

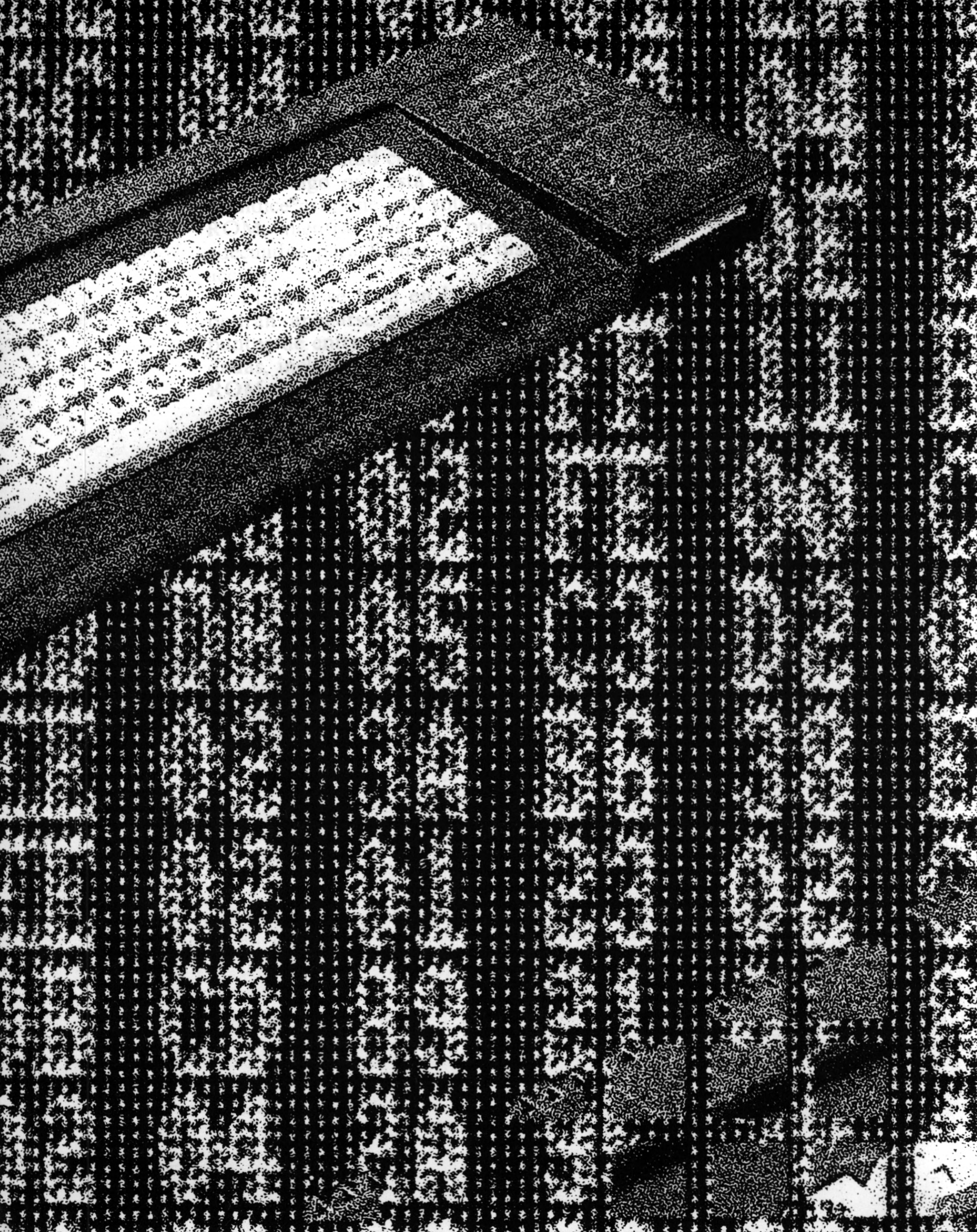
Una omisión notable es la no existencia de rutinas para la entrada y salida de datos numéricos que sean manejados por el intérprete principal. El proceso de entrada puede resultar complicado por la necesidad de cubrir bases binarias, decimales y hexadecimales, y por el hecho de que también puede tratarse de datos alfabéticos. Sin embargo, en términos amplios, el proceso comprende:

a) Comprobar si los datos son numéricos.
b) Conversión a valores binarios desde el ASCII.
c) Multiplicación del número introducido por la base del número.
d) Suma el nuevo dígito.
e) Enlazar con A.

La salida del bucle suele depender del hallazgo de un carácter no numérico en la fase *a*).

El proceso de salida es más difícil, sobre todo si debe incluirse el formato con exponente. Suele ser preferible colocar el número con cuidado, para lo cual necesitas saber el número de dígitos decimales y, a partir de ahí, el número de ceros que queden. El proceso implica la división por la base del número, tomando el resto como base para el dígito que va a mostrarse, pero esto produce primero el último dígito y debe juntarse una cadena de códigos en orden inverso antes de que la salida pueda comenzar.

El acceso a las rutinas de coma flotante abre la puerta a muchos tipos de programas en código máquina, que de otro modo resultarían mucho más difíciles de escribir, pero ello no quiere decir que tales programas pasen a ser fácilmente construibles. Hay que pensar mucho para que todo funcione bien, pero los resultados pueden ser muy satisfactorios.



14
El intérprete de BASIC

De existir debilidades en el sistema CPC464 y CPC6128, éstas se encuentran principalmente en el intérprete de BASIC. Aunque forma parte de la ROM junto al sistema operativo, la imagen que da es algo distinta. Hay algunas trabas innegables, que afortunadamente tienen un efecto mínimo sobre el funcionamiento global, 6 arrastra anidamiento de subrutinas hasta longitudes extremas, lo que hace las explicaciones resulten difíciles.

La mayor parte del intérprete está relacionada con la ejecución de los procesos relativos a palabras clave. La lista de palabras clave, señales y direcciones de entrada que se proporciona a continuación, simplificará la exploración, pero sólo son de aplicación para la versión 1.0. La extracción de la información correspondiente a otras versiones no es fácil, ya que los datos están muy dispersos.

En primer lugar está la tabla de palabras reservadas, que se organiza de modo ingenioso, aunque también confuso. De hecho, hay una serie de tablas cortas, una para cada letra del alfabeto. Si, por ejemplo, se va a buscar la palabra clave PRINT, la rutina conectará primero con el examen de la lista para la "P" y luego buscará "RINT". El último carácter de esta palabra se marca por la adición del bit 7, y el siguiente byte proporciona la señal para PRINT=&BF. La entrada es, en realidad, la primera de la lista de la "P", por lo que se encuentra enseguida; pero hay sólo 9 palabras en la lista, por lo que incluso la última es hallada rápidamente. Los sistemas que utilizan sólo una lista tardan mucho más en encontrar una entrada.

Una vez halladas las señales, necesitarás las direcciones de entrada; para esto has de tener paciencia, porque las tablas adecuadas están algo

diseminadas y toman formas diferentes. Para las señales &F4-&FD, tienen la forma de un bloque de saltos, reconocible por repetidas entradas &C3. Esto falla en CF81 para la versión 1.0. Se establecen otros enlaces mediante tablas especiales que relacionan señales con direcciones; otras tablas sólo dan las direcciones del enlace, accediéndose a las mismas partiendo de un puntero desplazado respecto a una base. Me temo que la única solución es buscar áreas de entradas no codificadas, y entonces hacer el trabajo de un pequeño detective para descubrir qué significan.

En conjunto, los puntos de entrada dentro del intérprete son de escaso interés para el programador de código máquina, sobre todo porque tienen acceso directo a las rutinas para matemáticas. Una excepción importante es la función CALL, que puede pasar parámetros a un programa en código máquina.

Cada parámetro es introducido como un número de dos bytes, que puede expresar un entero, un entero derivado de un número real (CF), o la dirección de un número real. El registro A se carga con el número de parámetros introducidos y IX apunta al último parámetro, de forma que, si hay N parámetros, el parámetro X se almacena en (N-X)*2, relativo a la dirección en IX.

A pesar de los comentarios hechos al principio de esta sección, hay muchos puntos de interés para cualquiera que explore el intérprete, pero examinarlos todos aquí ocuparía demasiado espacio. Dispones de los medios necesarios para la exploración, así que ¿por qué no los utilizas?

Palabras reservadas en orden de clave

Clave	Palabra	Dirección	Clave	Palabra	Dirección
&ØØ	ABS	FD85	&12	PEEK	F158
&Ø1	ASC	FA1Ø	&13	REMAIN	C99F
&Ø2	ATN	D53E	&14	SGN	FFØ2
&Ø3	CHR$	FA16	&15	SIN	D52F
&Ø4	CINT	FE8D	&16	SPACE$	FA57
&Ø5	COS	D534	&17	SP	D329
&Ø6	CREAL	FEEC	&18	SQR	D4EF
&Ø7	EXP	D52Ø	&19	STR$	F91E
&Ø8	FIX	FDE8	&1A	TAN	D539
&Ø9	FRE	FC2D	&1B	UNT	FEC2
&ØA	INKEY	D4Ø9	&1C	UPPER$	F842
&ØB	INP	F16D	&1D	VAL	FA77
&ØC	INT	FDED	&1E	-	FFØ6
&ØD	JOY	D423	$4Ø	EOF	C417
&ØE	LEN	FAØA	&41	ERR	DØDC
&ØF	LOG	D52A	&42	HIMEM	DØF4
&1Ø	LOG1Ø	D525	&43	INKEY$	FA24
&11	LOWER$	F834	&44	PI	D4ØB

Clave	Palabra	Dirección	Clave	Palabra	Dirección
&45	RND	D584	&9B	ERASE	D9ØØ
&46	TIME	DØE5	&9C	ERROR	CA8F
&47	XPOS	D1Ø7	&9D	EVERY	C979
&48	YPOS	D1ØE	&9E	FOR	C529
&71	BIN$	F8BA	&9F	GOSUB	C6ED
&72	DEC$	F8EA	&AØ	GOTO	C6E8
&73	HEX$	F8C4	&A1	IF	C6C7
&74	INSTR	FAA1	&A2	INC	C22A
&75	LEFT$	F93C	&A3	INPUT	DB2B
&76	MAX	D1EE	&A4	KEY	D439
&77	MIN	DIEA	&A5	LET	D654
&78	POS	C276	&A6	LINE	DAF8
&79	RIGHT$	F943	&A7	LIST	EØF7
&7A	ROUND	D219	&A8	LOAD	E9F6
&7B	STRING$	FA36	&A9	LOCATE	C2D2
&7C	TEST	C4E9	&AA	MEMORY	F4EF
&7D	TESTR	C4EE	&AB	MERGE	EAA6
&7E	–	CEAB	&AC	MID$	F993
&7F	VPOS	C262	&AD	MODE	C24F
&8Ø	AFTER	C971	&AE	MOVE	C5Ø5
&81	AUTO	CØDF	&AF	MOVER	C5ØA
&82	BORDER	C221	&BØ	NEXT	C5FB
&83	CALL	F1BA	&B1	NEW	C12B
&84	CAT	D246	&B2	ON	C7E3
&85	CHAIN	EA3C	&B3	ON BREAK	C8CB
&86	CLEAR	C132	&B4	ON ERROR	
&87	CLG	C485		GOTO	CBF8
&88	CLOSEIN	D298	&B5	ON SQ	C94Ø
&89	CLOSEOUT	D2A1	&B6	OPENIN	D25F
&8A	CLS	C25A	&B7	OPENOUT	D256
&8B	CONT	CBCØ	&B8	ORIGIN	C48C
&8C	DATA	E8EF	&B9	OUT	F177
&8D	DEF	D117	&BA	PAPER	C2ØA
&8E	DEFINT	D618	&BB	PEN	C212
&8F	DEFREAL	D61C	&BC	PLOT	C4DØ
&9Ø	DEFSTR	D614	&BD	PLOTR	C4D5
&91	DEG	D4E7	&BE	POKE	F15F
&92	DELETE	E728	&BF	PRINT	F1FD
&93	DIM	D67D	&CØ	–	E8F3
&94	DRAW	C4C6	&C1	RAD	D4EB
&95	DRAWR	C4CB	&C2	RANDOMISE	D55E
&96	EDIT	CØ52	&C3	READ	DCEB
&97	ELSE	E8F3	&C4	RELEASE	D31E
&98	END	CB65	&C5	REM	E8F3
&99	ENT	D385	&C6	RENUM	E7DF
&9A	ENV	D84E	&C7	RESTORE	DCD9
&C8	RESUME	CCØ3	&E4	FN	–
&C9	RETURN	C7ØF	&E5	SPC	–
&CA	RUN	E9BD	&E6	STEP	–
&CB	SAVE	ECØ9	&E7	SWAP	–
&CC	SOUND	D2CØ	&EA	TAB	–
&CD	SPEED	D494	&EB	THEN	–

Clave	Palabra	Dirección	Clave	Palabra	Dirección
&CE	STOP	CB5A	&EC	TO	-
&CF	SYMBOL	F69D	&ED	USING	-
&D0	TAG	C319	&EF	=	-
&D1	TAGOFF	C320	&F1	<	-
&D2	TROFF	DDE6	&F4	+	FCCC
&D3	TRON	DDE2	&F5	MINUS	FCE1
&D4	WAIT	F17D	&F6	*	FCF5
&D5	WEND	C776	&F7	/	FD12
&D6	WHILE	C747	&F8	-	D4F4
&D7	WIDTH	C3E3	&F9	DIV	FD37
&D8	WINDOW	C2E1	&FA	AND	FD58
&D9	WRITE	F47B	&FB	MOD	FD49
&DA	ZONE	F1F6	&FC	OR	FD63
&DB	DI	C8E1	&FD	XOR	FD6D
&DC	EI	C8E7	&FE	NOT	-
&E3	ERL	-			

No todas las claves están asociadas con direcciones. STEP será reconocido sólo por la rutina de FOR, mientras que SPC y USING se detectarán por PRINT.

Análogamente, algunas claves tienen un significado especial, que no está asociado con ninguna palabra. Por ejemplo, &FF previene que el siguiente byte es la clave de una función. Algunos bytes, que parecen ser claves, significan cosas muy diferentes. &02 introduce una variable entera; &0D introduce una variable real, y &1D introduce una dirección. &1E introduce una constante entera de dos bytes, y &1F, una constante real.

Es interesante e instructivo examinar un programa almacenado en memoria, que comienza a partir de la dirección 0170 en adelante, y comparar los códigos con la lista ofrecida anteriormente. Esto te dirá más de lo que pudieran transmitirte cien páginas de explicaciones.

Apéndice A
Programas de soporte

Los programas aquí dados te ayudarán en la exploración del sistema operativo y del intérprete de BASIC. El primero es un desensamblador que operará con el código de la RAM del ordenador. También realiza volcados hexadecimales y en forma de código ASCII. La presentación se puede enviar a la pantalla o a una impresora.

Se observará que se introduce una línea en blanco al final de cada bloque de código, lo cual ayudará a identificar las inserciones de datos. Escribir el programa puede ser un trabajo pesado, pero los resultados que obtendrás bien merecen la pena.

```
10 MEMORY &A4FF
20 GOSUB 4480
30 PF=0:PC=0
40 CLS
50 PRINT TAB(10);"1.- Modo pantalla."
60 PRINT TAB(10);"2.- Modo impresora."
70 PRINT TAB(10);"3.- Desensamblado."
80 PRINT TAB(10);"4.- Volcado hexadecimal."
90 PRINT TAB(10);"5.- Volcado de caracteres."
100 PRINT:PRINT TAB(15);
110 INPUT "< Selecciona opcion >",K
120 IF K<1 OR K>5 THEN 40
130 ON K GOTO 140,150,160,170,180
140 PF=0:GOTO 40
150 PF=8:GOTO 40
```

```
160 DF=0:GOTO 190
170 DF=1:GOTO 190
180 DF=2
190 CLS
200 INPUT "Direccion inicial ",C
210 INPUT "Direccion final ",E
220 C=C-65536*(C<0)
230 E=E-65536*(E<0)
240 IF E<C THEN 40
250 C=C-1
260 IF DF=0 THEN 360
270 PRINT #PF,:GOSUB 4540:PRINT #PF,HEX$(C+1,4)
280 GOSUB 4410
290 IF B<&20 OR B>&7F THEN BB=&3F ELSE BB=B
300 P=INT(C/(8*DF)):Q=C-(8*DF*P)
310 IF Q=0 THEN PRINT #PF:GOSUB 4540:PRINT #PF,HEX$(C,4);
320 IF DF=1 THEN X$=HEX$(B,2) ELSE X$=CHR$(BB)
330 PRINT #PF,TAB(6+(4-DF)*Q);X$;
340 IF C >= E THEN PRINT #PF:PRINT #PF: PC=PC+1 :GOSUB 4540:
    INPUT C:GOTO 40
350 GOTO 280
360 GOSUB 4410
370 NA=B
380 OS$=HEX$(C,4)+"  "+HEX$(B,2)+" "
390 A$=""
400 MA=NA\64:MD=NA MOD 64
410 ON (MA+1) GOTO 420,1210,1280,1390
420 MA=NA\8:MB=NA MOD 8
430 ON (MA+1) GOTO 440,530,620,710,800,890,980,1070
440 ON (MB+1) GOTO 450,460,470,480,490,500,510,520
450 A$="NOP":GOTO 1160
460 A$="LD   BC,":GOSUB 4130:GOTO 1160
470 A$="LD   (BC),A":GOTO 1160
480 A$="INC  BC":GOTO 1160
490 A$="INC  B":GOTO 1160
500 A$="DEC  B":GOTO 1160
510 A$="LD   B,":GOSUB 4080:GOTO 1160
520 A$="RLC  A":GOTO 1160
530 ON (MB+1) GOTO 540,550,560,570,580,590,600,610
540 A$="EX   AF,AF'":GOTO 1160
550 A$="ADD  HL,BC":GOTO 1160
560 A$="LD   A,(BC)":GOTO 1160
570 A$="DEC  BC":GOTO 1160
580 A$="INC  C":GOTO 1160
590 A$="DEC  C":GOTO 1160
600 A$="LD   C,":GOSUB 4080:GOTO 1160
610 A$="RRC  A":GOTO 1160
620 ON (MB+1) GOTO 630,640,650,660,670,680,690,700
630 A$="DJNZ ":GOSUB 4200:GOTO 1160
640 A$="LD   DE,":GOSUB 4130:GOTO 1160
650 A$="LD   (DE),A":GOTO 1160
660 A$="INC  DE":GOTO 1160
670 A$="INC  D":GOTO 1160
680 A$="DEC  D":GOTO 1160
```

```
690 A$="LD   D,":GOSUB 4080:GOTO 1160
700 A$="RL   A":GOTO 1160
710 ON (MB+1) GOTO 720,730,740,750,760,770,780,790
720 A$="JR   ":GOSUB 4200:FF=1:GOTO 1160
730 A$="ADD  HL,DE":GOTO 1160
740 A$="LD   A,(DE)":GOTO 1160
750 A$="DEC  DE":GOTO 1160
760 A$="INC  E":GOTO 1160
770 A$="DEC  E":GOTO 1160
780 A$="LD   E,":GOSUB 4080:GOTO 1160
790 A$="RR   A":GOTO 1160
800 ON (MB+1) GOTO 810,820,830,840,850,860,870,880
810 A$="JR   NZ,":GOSUB 4200:GOTO 1160
820 A$="LD   HL,":GOSUB 4130:GOTO 1160
830 A$="LD   (":GOSUB 4130:A$=A$+"),HL":GOTO 1160
840 A$="INC  HL":GOTO 1160
850 A$="INC  H":GOTO 1160
860 A$="DEC  H":GOTO 1160
870 A$="LD   H,":GOSUB 4080:GOTO 1160
880 A$="DAA  ":GOTO 1160
890 ON (MB+1) GOTO 900,910,920,930,940,950,960,970
900 A$="JR   Z,":GOSUB 4200:GOTO 1160
910 A$="ADD  HL,HL":GOTO 1160
920 A$="LD   HL,(":GOSUB 4130:A$=A$+")":GOTO 1160
930 A$="DEC  HL":GOTO 1160
940 A$="INC  L":GOTO 1160
950 A$="DEC  L":GOTO 1160
960 A$="LD   L,":GOSUB 4080:GOTO 1160
970 A$="CPL  ":GOTO 1160
980 ON (MB+1) GOTO 990,1000,1010,1020,1030,1040,1050,1060
990 A$="JR   NC,":GOSUB 4200:GOTO 1160
1000 A$="LD   SP,":GOSUB 4130:GOTO 1160
1010 A$="LD   (":GOSUB 4130:A$=A$+"),A":GOTO 1160
1020 A$="INC  SP":GOTO 1160
1030 A$="INC  (HL)":GOTO 1160
1040 A$="DEC  (HL)":GOTO 1160
1050 A$="LD   (HL),":GOSUB 4080:GOTO 1160
1060 A$="SCF  ":GOTO 1160
1070 ON (MB+1) GOTO 1080,1090,1100,1110,1120,1130,1140,1150
1080 A$="JR   C,":GOSUB 4200:GOTO 1160
1090 A$="ADD  HL,SP":GOTO 1160
1100 A$="LD   A,(":GOSUB 4130:A$=A$+")":GOTO 1160
1110 A$="DEC  SP":GOTO 1160
1120 A$="INC  A":GOTO 1160
1130 A$="DEC  A":GOTO 1160
1140 A$="LD   A,":GOSUB 4080:GOTO 1160
1150 A$="CCF  ":GOTO 1160
1160 PRINT #PF,OS$;TAB(19);A$
1170 IF FF<>0 THEN FF=0:GOSUB 4540
1180 GOSUB 4540
1190 IF C>=E THEN INPUT F:GOTO 40
1200 GOTO 360
1210 IF NA<>118 THEN 1230
1220 A$="HALT":GOTO 1270
```

```
1230 A$="LD   "
1240 MA=MD\8:MB=MD MOD 8
1250 ME=MA
1260 GOSUB 4320:A$=A$+",":ME=MB:GOSUB 4320:GOTO 1270
1270 GOTO 1160
1280 MA=MD\8:MB=MD MOD 8
1290 ON (MA+1) GOTO 1300,1310,1320,1330,1340,1350,1360,1370
1300 A$="ADD  A,":GOTO 1380
1310 A$="ADC  ":GOTO 1380
1320 A$="SUB  ":GOTO 1380
1330 A$="SBC  ":GOTO 1380
1340 A$="AND  ":GOTO 1380
1350 A$="XOR  ":GOTO 1380
1360 A$="OR   ":GOTO 1380
1370 A$="CP   ":GOTO 1380
1380 ME=MB:GOSUB 4320:GOTO 1160
1390 MA=MD\8:MB=MD MOD 8
1400 ON (MA+1) GOTO 1410,1500,1590,1680,1770,1860,1950,2040
1410 ON (MB+1) GOTO 1420,1430,1440,1450,1460,1470,1480,1490
1420 A$="RET  NZ":GOTO 1160
1430 A$="POP  BC":GOTO 1160
1440 A$="JP   NZ,":GOSUB 4130:GOTO 1160
1450 A$="JP   ":GOSUB 4130:FF=1:GOTO 1160
1460 A$="CALL NZ,":GOSUB 4130:GOTO 1160
1470 A$="PUSH BC":GOTO 1160
1480 A$="ADD  A,":GOSUB 4080:GOTO 1160
1490 A$="RESET ":FF=1:GOTO 1160
1500 ON (MB+1) GOTO 1510,1520,1530,1540,1550,1560,1570,1580
1510 A$="RET  Z":GOTO 1160
1520 A$="RET  ":FF=1:GOTO 1160
1530 A$="JP   Z,":GOSUB 4130:GOTO 1160
1540 GOTO 2140
1550 A$="CALL Z,":GOSUB 4130:GOTO 1160
1560 A$="CALL ":GOSUB 4130:GOTO 1160
1570 A$="ADC  ":GOSUB 4080:GOTO 1160
1580 A$="INF SALTO ":FF=1:GOSUB 4130:GOTO 1160
1590 ON (MB+1) GOTO 1600,1610,1620,1630,1640,1650,1660,1670
1600 A$="RET  NC":GOTO 1160
1610 A$="POP  DE":GOTO 1160
1620 A$="JP   NC,":GOSUB 4130:GOTO 1160
1630 A$="OUT  (":GOSUB 4080:A$=A$+",A":GOTO 1160
1640 A$="CALL NC,":GOSUB 4130:GOTO 1160
1650 A$="PUSH DE":GOTO 1160
1660 A$="SUB  ":GOSUB 4080:GOTO 1160
1670 A$="LLAMADA LATERAL ":GOSUB 4130:GOTO 1160
1680 ON (MB+1) GOTO 1690,1700,1710,1720,1730,1740,1750,1760
1690 A$="RET  C":GOTO 1160
1700 A$="EXX   ":GOTO 1160
1710 A$="JP   C,":GOSUB 4130:GOTO 1160
1720 A$="IN   A,(":GOSUB 4080:A$=A$+")":GOTO 1160
1730 A$="CALL C,":GOSUB 4130:GOTO 1160
1740 A$="IX":GOTO 2330
1750 A$="SBC  ":GOSUB 4080:GOTO 1160
1760 A$="LLAMADA ":GOSUB 4130:GOTO 1160
```

```
1770 ON (MB+1) GOTO 1780,1790,1800,1810,1820,1830,1840,1850
1780 A$="RET  PO":GOTO 1160
1790 A$="POP  HL":GOTO 1160
1800 A$="JP   PO,":GOSUB 4130:GOTO 1160
1810 A$="EX   (SP),HL":GOTO 1160
1820 A$="CALL PO,":GOSUB 4130:GOTO 1160
1830 A$="PUSH HL":GOTO 1160
1840 A$="AND  ":GOSUB 4080:GOTO 1160
1850 A$="LAM RAM ":GOTO 1160
1860 ON (MB+1) GOTO 1870,1880,1890,1900,1910,1920,1930,1940
1870 A$="RET  PE":GOTO 1160
1880 A$="JP   (HL)":FF=1:GOTO 1160
1890 A$="JP   PE":GOTO 1160
1900 A$="EX   DE,HL":GOTO 1160
1910 A$="CALL PE,":GOSUB 4130:GOTO 1160
1920 GOTO 3180
1930 A$="XOR  ":GOSUB 4080:GOTO 1160
1940 A$="SALTO ROM INFERIOR ":FF=1:GOSUB 4130:GOTO 1160
1950 ON (MB+1) GOTO 1960,1970,1980,1990,2000,2010,2020,2030
1960 A$="RET  P":GOTO 1160
1970 A$="POP  AF":GOTO 1160
1980 A$="JP   P,":GOSUB 4130:GOTO 1160
1990 A$="DI   ":GOTO 1160
2000 A$="CALL P,":GOSUB 4130:GOTO 1160
2010 A$="PUSH AF":GOTO 1160
2020 A$="OR   ":GOSUB 4080:GOTO 1160
2030 A$="RST USUARIO ":FF=1:GOTO 1160
2040 ON (MB+1) GOTO 2050,2060,2070,2080,2090,2100,2110,2120
2050 A$="RET  M":GOTO 1160
2060 A$="LD   SP,HL":GOTO 1160
2070 A$="JP   M,":GOSUB 4130:GOTO 1160
2080 A$="EI   ":GOTO 1160
2090 A$="CALL M,":GOSUB 4130:GOTO 1160
2100 A$="IY":GOTO 2330
2110 A$="CP   ":GOSUB 4080:GOTO 1160
2120 A$="INTERRUPCION ":GOTO 1160
2130 GOTO 1160
2140 GOSUB 4410:RE=B:OS$=OS$+HEX$(B,2)+" "
2150 MA=RE\64:MD=RE MOD 64
2160 ON (MA+1) GOTO 2170,2270,2290,2310
2170 MC=MD\8:MB=MD MOD 8
2180 ON (MC+1) GOTO 2190,2200,2210,2220,2230,2240,2260,2260
2190 A$="RLC  ":ME=MB:GOSUB 4320:GOTO 2130
2200 A$="RRC  ":ME=MB:GOSUB 4320:GOTO 2130
2210 A$="RL   ":ME=MB:GOSUB 4320:GOTO 2130
2220 A$="RR   ":ME=MB:GOSUB 4320:GOTO 2130
2230 A$="SLA  ":ME=MB:GOSUB 4320:GOTO 2130
2240 A$="SRA  ":ME=MB:GOSUB 4320:GOTO 2130
2250 GOTO 4300
2260 A$="SRL  ":ME=MB:GOSUB 4320:GOTO 2130
2270 MC=MD\8:MB=MD MOD 8:ME=MB
2280 A$="BIT "+STR$(MC)+",":GOSUB 4320:GOTO 2130
2290 MC=MD\8:MB=MD MOD 8:ME=MB
2300 A$="RES "+STR$(MC)+",":GOSUB 4320:GOTO 2130
```

```
2310 MC=MD\8:MB=MD MOD 8:ME=MB
2320 A$="SET "+STR$(MC)+",":GOSUB 4320:GOTO 2130
2330 GOSUB 4410:RE=B:OS$=OS$+HEX$(RE,2)+" "
2340 MA=RE\64:MD=RE MOD 64
2350 ON (MA+1) GOTO 2360,2610,2790,2970
2360 MC=MD\8:MB=MD MOD 8
2370 ON (MC+1) GOTO 2380,2390,2410,2420,2440,2490,2540,2590
2380 GOTO 4300
2390 IF MB<>1 THEN 4300
2400 A$="ADD  "+AX$+",BC":GOTO 2130
2410 GOTO 4300
2420 IF MB<>1 THEN 4300
2430 A$="ADD  "+AX$+",DE":GOTO 2130
2440 IF MB>3 THEN 4300
2450 ON (MB+1) GOTO 4300,2460,2470,2480
2460 A$="LD   "+AX$+",":GOSUB 4130:GOTO 2130
2470 A$="LD   (":GOSUB 4130:A$=A$+"),"+AX$:GOTO 2130
2480 A$="INC  "+AX$:GOTO 2130
2490 IF MB=0 OR MB>3 THEN 4300
2500 ON MB GOTO 2510,2520,2530
2510 A$="ADD  "+AX$+","+AX$:GOTO 2130
2520 A$="LD   "+AX$+",(":GOSUB 4130:A$=A$+")":GOTO 2130
2530 A$="DEC  "+AX$:GOTO 2130
2540 IF MB<4 OR MB=7 THEN 4300
2550 ON (MB-3) GOTO 2560,2570,2580
2560 A$="INC  ":GOSUB 4310:GOTO 2130
2570 A$="DEC  ":GOSUB 4310:GOTO 2130
2580 A$="LD   ":GOSUB 4310:A$=A$+",":GOSUB 4080:GOTO 2130
2590 IF MB<>1 THEN 4300
2600 A$="ADD  "+AX$+",SP":GOTO 2130
2610 MC=MD\8:MB=MD MOD 8
2620 ON (MC+1) GOTO 2630,2650,2670,2690,2710,2730,2750,2770
2630 IF MB<>6 THEN 4300
2640 A$="LD   B,":GOSUB 4310:GOTO 2130
2650 IF MB<>6 THEN 4300
2660 A$="LD   C,":GOSUB 4310:GOTO 2130
2670 IF MB<>6 THEN 4300
2680 A$="LD   D,":GOSUB 4310:GOTO 2130
2690 IF MB<>6 THEN 4300
2700 A$="LD   E,":GOSUB 4310:GOTO 2130
2710 IF MB<>6 THEN 4300
2720 A$="LD   H,":GOSUB 4310:GOTO 2130
2730 IF MB<>6 THEN 4300
2740 A$="LD   L,":GOSUB 4310:GOTO 2130
2750 IF MB=6 THEN 4300
2760 A$="LD   ":GOSUB 4310: A$=A$+",": ME=MB:GOSUB 4320:GOTO
     2130
2770 IF MB<>6 THEN 4300
2780 A$="LD   A,":GOSUB 4310:GOTO 2130
2790 MC=MD\8:MB=MD MOD 8
2800 ON (MC+1) GOTO 2810,2830,2850,2870,2890,2910,2930,2950
2810 IF MB<>6 THEN 4300
2820 A$="ADD  A,":GOSUB 4310:GOTO 2130
2830 IF MB<>6 THEN 4300
```

```
2840 A$="ADC  A,":GOSUB 4310:GOTO 2130
2850 IF MB<>6 THEN 4300
2860 A$="SUB  A,":GOSUB 4310:GOTO 2130
2870 IF MB<>6 THEN 4300
2880 A$="SBC  A,":GOSUB 4310:GOTO 2130
2890 IF MB<>6 THEN 4300
2900 A$="AND  A,":GOSUB 4310:GOTO 2130
2910 IF MB<>6 THEN 4300
2920 A$="XOR  A,":GOSUB 4310:GOTO 2130
2930 IF MB<>6 THEN 4300
2940 A$="OR   A,":GOSUB 4310:GOTO 2130
2950 IF MB<>6 THEN 4300
2960 A$="CP   A,":GOSUB 4310:GOTO 2130
2970 MC=MD\8:MB=MD MOD 8
2980 ON (MC+1) GOTO 4300,2990,4300,4300,3960,4000,4300,4040
2990 IF MB<>3 THEN 4300
3000 GOSUB 4410:NC=B:OS$=OS$+HEX$(B,2)+"  "
3010 GOSUB 4410:RH=B:OS$=OS$+HEX$(B,2)+"  "
3020 IF (RH-6) MOD 8<>0 THEN 4300
3030 MC=RH\64:MB=RH\8
3040 ON (MC+1) GOTO 3050,3130,3140,3150
3050 ON (MB+1) GOTO 3060,3070,3080,3090,3100,3110,4300,3120
3060 A$="RLC  ":GOSUB 3160:GOTO 1160
3070 A$="RRC  ":GOSUB 3160:GOTO 1160
3080 A$="RL   ":GOSUB 3160:GOTO 1160
3090 A$="RR   ":GOSUB 3160:GOTO 1160
3100 A$="SLA  ":GOSUB 3160:GOTO 1160
3110 A$="SRA  ":GOSUB 3160:GOTO 1160
3120 A$="SRL  ":GOSUB 3160:GOTO 1160
3130 A$="BIT  ":GOSUB 3170:GOTO 1160
3140 A$="RES  ":GOSUB 3170:GOTO 1160
3150 A$="SET  ":GOSUB 3170:GOTO 1160
3160 A$=A$+"("+AX$+"+"+HEX$(NC)+")":RETURN
3170 A$=A$+HEX$((RH AND &38)/8)+",("+AX$+"+"+HEX$(NC)+")":
     RETURN
3180 GOSUB 4410:RE=B:OS$=OS$+HEX$(B,2)+"  "
3190 MA=RE\64:MB=RE MOD 64
3200 ON (MA+1) GOTO 4300,3210,3730,4300
3210 MD=MB\8:MB=MB MOD 8
3220 ON (MD+1) GOTO 3230,3320,3390,3460,3530,3590,3650,3680
3230 ON (MB+1) GOTO 3240,3250,3260,3270,3280,3300,3310,3310
3240 A$="IN   B,(C)":GOTO 1160
3250 A$="OUT  (C),B":GOTO 1160
3260 A$="SBC  HL,BC":GOTO 1160
3270 A$="LD   (":GOSUB 4130:A$=A$+"),BC":GOTO 1160
3280 A$="NEG  ":GOTO 1160
3290 A$="RET  N":GOTO 1160
3300 A$="IM   0":GOTO 1160
3310 A$="LD   I,A":GOTO 1160
3320 ON (MB+1) GOTO 3330,3340,3350,3360,4300,3370,3400,3380
3330 A$="IN   C,(C)":GOTO 1160
3340 A$="OUT  (C),C":GOTO 1160
3350 A$="ADC  HL,BC":GOTO 1160
3360 A$="LD   BC,(":GOSUB 4130:A$=A$+")":GOTO 1160
```

```
3370 A$="RET  I":GOTO 1160
3380 A$="LD   R,A":GOTO 1160
3390 ON (MB+1) GOTO 3400,3410,3420,3430,4300,4300,3440,3450
3400 A$="IN   D,(C)":GOTO 1160
3410 A$="OUT  (C),D":GOTO 1160
3420 A$="SBC  HL,DE":GOTO 1160
3430 A$="LD   (":GOSUB 4130:A$=A$+"),DE":GOTO 1160
3440 A$="IM   1":GOTO 1160
3450 A$="LD   A,I":GOTO 1160
3460 ON (MB+1) GOTO 3470,3480,3490,3500,4300,4300,3510,3520
3470 A$="IN   E,(C)":GOTO 1160
3480 A$="OUT  (C),E":GOTO 1160
3490 A$="ADC  HL,DE":GOTO 1160
3500 A$="LD   DE,(":GOSUB 4130:A$=A$+")":GOTO 1160
3510 A$="IM   2":GOTO 1160
3520 A$="LD   A,R":GOTO 1160
3530 ON (MB+1) GOTO 3540,3550,3560,3570,4300,4300,4300,3580
3540 A$="IN   H,(C)":GOTO 1160
3550 A$="OUT  (C),H":GOTO 1160
3560 A$="SBC  HL,HL":GOTO 1160
3570 A$="LD   (":GOSUB 4130:A$=A$+"),HL":GOTO 1160
3580 A$="RRD  ":GOTO 1160
3590 ON (MB+1) GOTO 3600,3610,3620,3630,4300,4300,4300,3640
3600 A$="IN   L,(C)":GOTO 1160
3610 A$="OUT  (C),L":GOTO 1160
3620 A$="ADC  HL,HL":GOTO 1160
3630 A$="LD   HL,(":GOSUB 4130:A$=A$+")":GOTO 1160
3640 A$="RLD  ":GOTO 1160
3650 ON (MB+1) GOTO 4300,4300,3660,3670,4300,4300,4300,4300
3660 A$="SBC  HL,SP":GOTO 1160
3670 A$="LD   (":GOSUB 4130:A$=A$+"),SP":GOTO 1160
3680 ON (MB+1) GOTO 3690,3700,3710,3720,4300,4300,4300,4300
3690 A$="IN   A,(C)":GOTO 1160
3700 A$="OUT  (C),A":GOTO 1160
3710 A$="ADC  HL,SP":GOTO 1160
3720 A$="LD   SP,(":GOSUB 4130:A$=A$+")":GOTO 1160
3730 MD=MB\8:MB=MB MOD 8
3740 IF MD<4 OR MB>3 THEN 4300
3750 ON (MD-3) GOTO 3760,3810,3860,3910
3760 ON (MB+1) GOTO 3770,3780,3790,3800
3770 A$="LDI  ":GOTO 1160
3780 A$="CPI  ":GOTO 1160
3790 A$="INI  ":GOTO 1160
3800 A$="OUTI ":GOTO 1160
3810 ON (MB+1) GOTO 3820,3830,3840,3850
3820 A$="LDD  ":GOTO 1160
3830 A$="CPD  ":GOTO 1160
3840 A$="IND  ":GOTO 1160
3850 A$="OUTD ":GOTO 1160
3860 ON (MB+1) GOTO 3870,3880,3890,3900
3870 A$="LDIR ":GOTO 1160
3880 A$="CPIR ":GOTO 1160
3890 A$="INIR ":GOTO 1160
3900 A$="OTIR ":GOTO 1160
```

```
3910 ON (MB+1) GOTO 3920,3930,3940,3950
3920 A$="LDDR ":GOTO 1160
3930 A$="CPDR ":GOTO 1160
3940 A$="INDR ":GOTO 1160
3950 A$="OTDR ":GOTO 1160
3960 ON (MB+1) GOTO 4300,3970,4300,3980,4300,3990,4300,4300
3970 A$="POP  "+AX$:GOTO 1160
3980 A$="EX   (SP),"+AX$:GOTO 1160
3990 A$="PUSH "+AX$:GOTO 1160
4000 IF MB>1 THEN 4300
4010 ON (MB+1) GOTO 4020,4030
4020 A$="ADC  A,":GOSUB 4310:GOTO 1160
4030 A$="JP   ("+AX$+")":GOTO 1160
4040 IF MB<>1 THEN 4300
4050 A$="LD   SP,"+AX$:GOTO 1160
4060 IF MB<>1 THEN 4300
4070 A$="LD   SP,"+AX$:GOTO 1160
4080 A$=A$+"#":GOSUB 4410
4090 NC=B
4100 A$=A$+HEX$(NC,2)
4110 OS$=OS$+HEX$(NC,2)+"  "
4120 RETURN
4130 GOSUB 4410
4140 NC=B
4150 GOSUB 4410
4160 NE=B
4170 A$=A$+HEX$(NE,2)+HEX$(NC,2)
4180 OS$=OS$+HEX$(NC,2)+"  "+HEX$(NE,2)+"  "
4190 RETURN
4200 GOSUB 4410
4210 ND=B:OS$=OS$+HEX$(ND,2)+"  "
4220 IF ND>127 THEN 4260
4230 ND=ND+C+1
4240 A$=A$+HEX$(ND,4)
4250 RETURN
4260 ND=ND+C-255
4270 GOTO 4240
4280 RETURN
4290 RETURN
4300 A$="Codigo invalido":GOTO 1160
4310 A$=A$+"("+AX$+"+":GOSUB 4080:A$=A$+")":RETURN
4320 ON (ME+1) GOTO 4330,4340,4350,4360,4370,4380,4390,4400
4330 A$=A$+"B":RETURN
4340 A$=A$+"C":RETURN
4350 A$=A$+"D":RETURN
4360 A$=A$+"E":RETURN
4370 A$=A$+"H":RETURN
4380 A$=A$+"L":RETURN
4390 A$=A$+"(HL)":RETURN
4400 A$=A$+"A":RETURN
4410 C=C+1
4420 Q=INT(C/256)
4430 POKE &A614,Q
4440 POKE &A613,(C-256*Q)
```

```
4450 CALL &A600
4460 B=PEEK(&A615)
4470 RETURN
4480 FOR X=&A600 TO &A612
4490 READ Y:POKE X,Y:NEXT
4500 RETURN
4510 DATA &2A,&13,&A6,&CD,&00,&B9,&F5
4520 DATA &CD,&06,&B9,&7E,&32,&15,&A6
4530 DATA &F1,&CD,&0C,&B9,&C9
4540 IF PF<>8 THEN RETURN
4550 PC=PC+1
4560 IF PC<60 THEN RETURN
4570 FOR Y=1 TO 6
4580 PRINT #PF
4590 NEXT
4600 PC=PC-60
4610 RETURN
```

El segundo programa te permitirá llamar a rutinas concretas conociendo el contenido de los registros. El contenido de los registros a la salida también se recupera y será evaluado cualquier número en coma flotante contenido en DE y HL. El programa se debe manejar con cuidado, ya que algunas llamadas pueden bloquear de alguna manera al sistema si no se introducen los parámetros adecuados.

```
10 GOSUB 390
20 CLS
30 INPUT " Registro BC: ",BC
40 BC=BC-65536*(BC<0)
50 B=INT(BC/256):C=BC-B*256
60 POKE &4001,C:POKE &4002,B
70 INPUT " Registro DE: ",DE
80 DE=DE-65536*(DE<0)
90 D=INT(DE/256):E=DE-D*256
100 POKE &4004,E:POKE &4005,D
110 INPUT " Registro HL: ",HL
120 HL=HL-65536*(HL<0)
130 H=INT(HL/256):L=HL-H*256
140 POKE &4007,L:POKE &4008,H
150 INPUT " Registro A:  ",A
160 POKE &400A,A
170 INPUT " Direccion; CALL ",NM
180 NM=NM-65536*(NM<0)
190 N=INT(NM/256):M=NM-N*256
200 POKE &400C,M:POKE &400D,N
210 CALL &4000
220 BC=PEEK(&4020)+256*PEEK(&4021)
230 DE=PEEK(&4022)+256*PEEK(&4023)
240 HL=PEEK(&4024)+256*PEEK(&4025)
250 A=PEEK(&4026)
260 PRINT " BC=";HEX$(BC,4)
270 PRINT " DE=";HEX$(DE,4)
```

```
280 PRINT " HL=";HEX$(HL,4)
290 PRINT "  A=";HEX$(A,2)
300 R=DE
310 N$="DE"
320 GOSUB 470
330 R=HL
340 N$="HL"
350 GOSUB 470
360 PRINT
370 INPUT Y
380 GOTO 20
390 FOR P=&4000 TO &401F
400 READ Q:POKE P,Q
410 NEXT
420 RETURN
430 DATA 1,0,0,&11,0,0,&21,0
440 DATA 0,&3E,0,&CD,0,0,&ED,&43
450 DATA &20,&40,&ED,&53,&22,&40,&22,&24
460 DATA &40,&E5,&21,&26,&40,&77,&E1,&C9
470 IF ABS(R)<16384 THEN RETURN
480 PRINT "CF(";N$;")=";
490 FOR X=4 TO 0 STEP -1
500 PRINT HEX$(PEEK(R+X),2);
510 NEXT
520 PRINT " = ";
530 Z=0
540 FOR X=0 TO 3
550 Z=(Z+PEEK(R+X))/256
560 NEXT
570 IF Z<0.5 THEN Z=Z+0.5:PRINT "+"; ELSE PRINT "-";
580 Z=Z*2^(PEEK(R+4)-&80)
590 PRINT Z
600 RETURN
```

Apéndice B

Indice por direcciones

Este es el índice de las etiquetas utilizadas en este libro, para identificar las diferentes direcciones, subrutinas y vectores en los Amstrad CPC464 y CPC6128. Las referencias son direcciones en memoria y no números de páginas. Los números están dados en hexadecimal. El índice está en orden numérico.

Rutina	*Dirección*
ACTIVA ROM SUPERIOR	B900, BA5E
INHIBE ROM SUPERIOR	B903, BA68
ACTIVA ROM INFERIOR	B906, BA4A
INHIBE ROM INFERIOR	B909, BA54
RESTAURA ROM	B90C, BA72
SELECCIONA ROM	B90F, BA7E
ROM ACTUAL	B912, BAA2
TIPO ROM	B915, BAA2
ROM ANTERIOR	B918, BA8C
LDIR	B91B, BAA6
LDDR	B91E, BAAC
NC PRIORIDAD	B921,
INF SALTO HL	000B, B97C
INF SALTO	0008, B982
SALTO HL	001B, B9B1
LLAMADA HL	0023, B9B9

LLAMADA	0018, B9BF
SALTO HL LATERAL	0013, BA10
LLAMADA LATERAL	0010, BA16
SALTO ROM INFERIOR	0028, BA2E
LAM RAM	0020, BACB
TCL INICIALIZACION	BB00, 19E0
TCL REINICIALIZACION	BB03, 1A1E
TCL LEE CARACTER	BB09, 1A42
TCL ESPERA CARACTER	BB06, 1A3C
TCL REGRESA CARACTER	BB0C, 1A77
TCL PONE EXPANSION	BB0F, 1ABD
TCL TOMA EXPANSION	BB12, 1B2E
TCL ASIGNA BUFFER	BB15, 1A7B
TCL ESPERA TECLA	BB18, 1B56
TCL LEE TECLA	BB1B, 1B5C
TCL EXAMINA TECLA	BB1E, 1CBD
TCL TOMA ESTADO	BB21, 1BB3
TCL ESTADO JOYSTICK	BB24, 1C5C
TCL PONE TRASLADO	BB27, 1052
TCL TOMA TRASLADO	BB2A, 1D3E
TCL TRASLADO CON SHIFT	BB2D, 1D57
TCL OBTIENE SHIFT	BB30, 1D43
TCL TRASLADO CON CONTROL	BB33, 1D5C
TCL OBTIENE CONTROL	BB36, 1D48
TCL IMPONE REPETICION	BB39, 1CAB
TCL CONOCE REPETICION	BB3C, 1CA6
TCL IMPONE RETARDO	BB3F, 1C6D
TCL CONOCE RETARDO	BB42, 1C69
TCL PERMITE RUPTURAS	BB45, 1C71
TCL INHIBE RUPTURAS	BB48, 1C82
TCL GENERA RUPTURA	BB4B, 1C90
TXT INICIALIZA	BB4E, 1078
TXT REINICIALIZA	BB51, 1088
TXT ACTIVA VDU	BB54, 1415
TXT DESACTIVA VDU	BB57, 144B
TXT SALIDA	BB5A, 1400
TXT ESCRIBE CARACTER	BB5D, 1334
TXT LEE CARACTER	BB60, 13AB
TXT ADMISION DE GRAFICOS	BB63, 137A
TXT ACTIVA VENTANA	BB66, 120C
TXT EXAMINA VENTANA	BB69, 1256
TXT LIMPIA VENTANA	BB6C, 1540
TXT FIJA COLUMNA	BB6F, 115E
TXT FIJA FILA	BB72, 1174
TXT FIJA CURSOR	BB75, 1174
TXT EXAMINA CURSOR	BB78, 1180

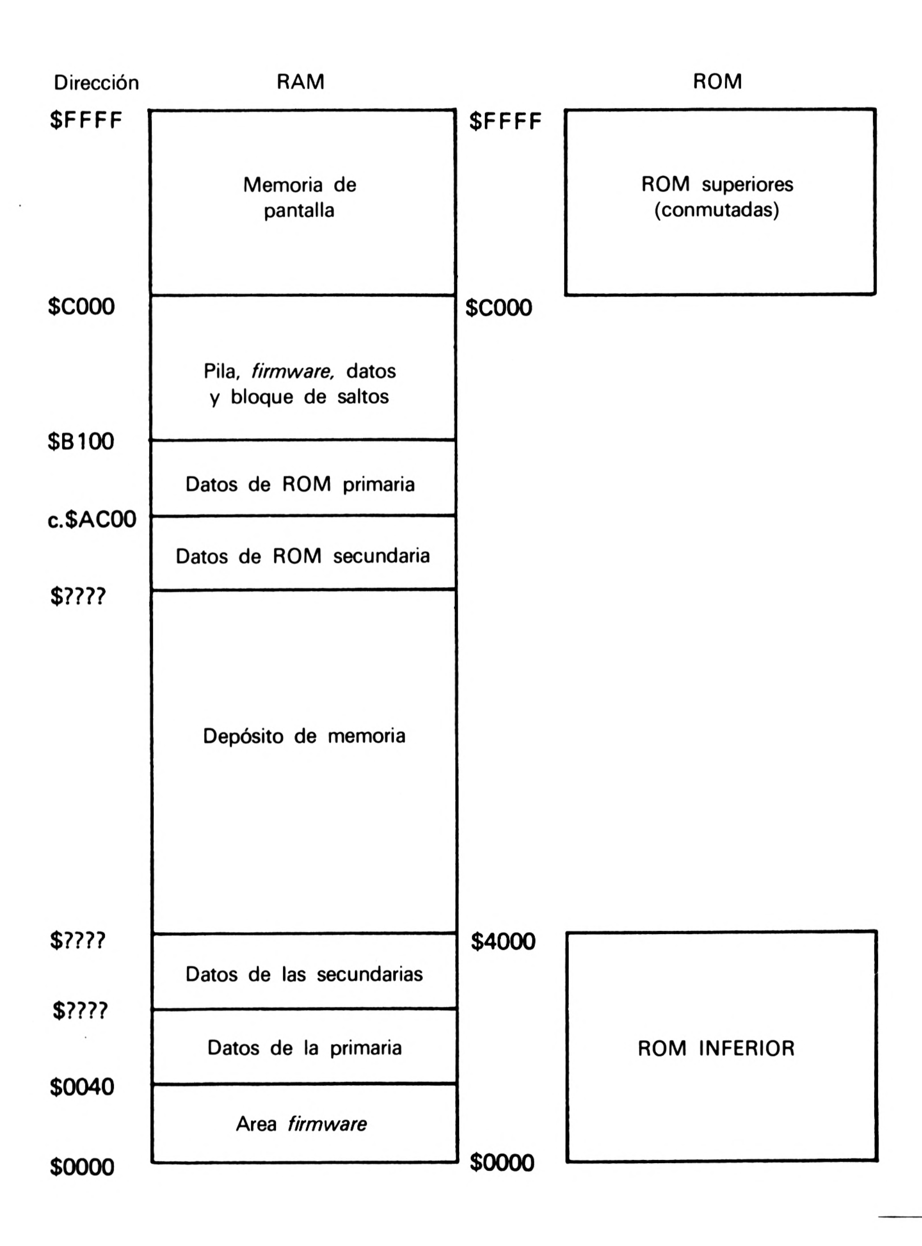
TXT ACTIVA CURSOR USUARIO	BB7B, 1289
TXT INHIBE CURSOR USUARIO	BB7E, 129A
TXT ACTIVA CURSOR SISTEMA	BB81, 1279
TXT INHIBE CURSOR SISTEMA	BB84, 1281
TXT POSICION VALIDA	BB87, 11CE
TXT PONE CURSOR	BB8A, 1268
TXT QUITA CURSOR	BB8D, 1268
TXT COLOR PLUMA	BB90, 12A9
TXT EXAMINA PLUMA	BB93, 12BD
TXT COLOR PAPEL	BB96, 12AE
TXT EXAMINA PAPEL	BB99, 12C3
TXT INVIERTE COLORES	BB9C, 12C9
TXT FIJA MODO	BB9F, 137A
TXT EXAMINA MODO	BBA2, 1387
TXT EXAMINA MATRIZ	BBA5, 12D3
TXT IMPONE MATRIZ	BBA8, 12F1
TXT FIJA TABLA MATRICES	BBAB, 12FD
TXT EXAMINA TABLA MATRICES	BBAE, 132A
TXT EXAMINA TABLA CONTROL	BBB1, 14CB
TXT SELECCIONA CAUCE	BBB4, 10E8
TXT CANJEA CAUCES	BBB7, 1107
GRA INICIALIZA	BBBA, 15B0
GRA REINICIALIZA	BBBD, 15DF
GRA MOV ABSOLUTO	BBC0, 15F4
GRA MOV RELATIVO	BBC3, 15F1
GRA POSICION CURSOR	BBC6, 15FC
GRA FIJA ORIGEN	BBC9, 1604
GRA OBTIENE ORIGEN	BBCC, 1612
GRA ANCHURA VENTANA	BBCF, 1734
GRA ALTURA VENTANA	BBD2, 1779
GRA OBTIENE ANCHURA	BBD5, 17A6
GRA LIMPIA VENTANA	BBDB, 17C5
GRA COLOR PLUMA	BBDE, 17F6
GRA EXAMINA PLUMA	BBE1, 1804
GRA COLOR PAPEL	BBE4, 17FD
GRA EXAMINA PAPEL	BBE7, 1807
GRA PUNTO ABSOLUTO	BBEA, 1813
GRA PUNTO RELATIVO	BBED, 1810
GRA TEST PUNTO ABS	BBF0, 1827
GRA TEST PUNTO REL	BBF3, 1824
GRA LINEA ABSOLUTA	BBF6, 1839
GRA LINEA RELATIVA	BBF9, 1836
GRA ESCRIBE CARACTER	BBFC, 1945
PNT INICIALIZA	BBFF, 0AA0
PNT REINICIALIZA	BC02, 0AB1
PNT IMPONE OFFSET	BC05, 0B3C

PNT IMPONE BASE RAM	BC08, 0B45
PNT LOCALIZA RAM PANTALLA	BC0B, 0B50
PNT FIJA MODO	BC0E, 0ACA
PNT EXAMINA MODO	BC11, 0AEC
PNT BORRA PANTALLA	BC14, 0AF2
PNT LIMITES PANTALLA	BC17, 0B57
PNT POSICIONA COORDENADAS	BC1A, 0B64
PNT COORDENADA PUNTOS	BC1D, 0B95
PNT SIGUIENTE BYTE	BC20, 0BF9
PNT BYTE ANTERIOR	BC23, 0C05
PNT SIGUIENTE LINEA	BC26, 0C13
PNT LINEA ANTERIOR	BC29, 0C2D
PNT CODIFICA TINTA	BC2C, 0C86
PNT DECODIFICA TINTA	BC2F, 0CA0
PNT IMPONE TINTA	BC32, 0CEC
PNT OBTIENE TINTA	BC35, 0D14
PNT IMPONE BORDE	BC38, 0CF1
PNT OBTIENE BORDE	BC3B, 0D19
PNT FIJA FLASH	BC3E, 0CE4
PNT EXAMINA FLASH	BC41, 0CE8
PNT RELLENA CAJA	BC44, 0DB3
PNT RELLENA BYTE	BC47, 0DB7
PNT INVIERTE CARACTER	BC4A, 0DDF
PNT DESPLAZA PANTALLA	BC4D, 0DFA
PNT DESPLAZA AREA	BC50, 0E3E
PNT EXPANDE	BC53, 0EF3
PNT COMPRIME	BC56, 0F49
PNT MODO GRAFICOS	BC59, 0C49
PNT PIXEL	BC5C, 0C6B
PNT HORIZONTAL	BC5F, 0FC4
PNT VERTICAL	BC62, 102F
CAS INICIALIZA	BC65, 2370
CAS VELOCIDAD	BC68, 237F
CAS MENSAJES	BC6B, 238E
CAS ARRANCA MOTOR	BC6E, 2A4B
CAS PARA MOTOR	BC71, 2A4F
CAS RESTAURA MOTOR	BC74, 2A51
CAS ABRE ENTRADA	BC77, 2392
CAS CIERRA ENTRADA	BC7A, 23FC
CAS ABANDONA ENTRADA	BC7D, 2401
CAS LEE CARACTER	BC80, 2435
CAS ENTRADA DIRECTA	BC83, 24AB
CAS RETORNO	BC86, 249A
CAS EXAMINA FINAL	BC89, 2496
CAS ABRE SALIDA	BC8C, 23AB
CAS CIERRA SALIDA	BC8F, 2415

CAS ABANDONA SALIDA	BC92, 242E
CAS SALIDA CARACTER	BC95, 245B
CAS SALIDA DIRECTA	BC98, 24EA
CAS CATALOGA	BC9B, 2528
CAS ESCRIBE	BC9E, 283F
CAS LEE	BCA1, 2836
CAS COMPARA	BCA4, 2851
REINICIALIZA SONIDO	BCA7, 1E68
COLA SONIDO	BCAA, 1F9F
EXAMINA COLA	BCAD, 206C
FORMA SUCESO SONIDO	BCB0, 2089
EJECUTA SONIDO	BCB3, 204A
PARA SONIDO	BCB6, 1ECB
CONTINUA SONIDO	BCB9, 1EE6
ENVOLVENTE VOLUMEN	BCBC, 2338
ENVOLVENTE TONO	BCBF, 233D
DIRECCION ENVOLVENTE V	BCC2, 2349
DIRECCION ENVOLVENTE T	BCC5, 234E
NC REINICIALIZA NUCLEO	BCC8, 005C
NC ROM SECUNDARIA	BCCB, 0329
NC INIC SECUNDARIA	BCCE, 0332
NC INTRODUCE RSX	BCD1, 02A1
NC BUSCA ORDEN	BCD4, 02B2
NC NUEVO SINCRONO BARRIDO	BCD7, 0163
NC AÑADE SINCRONO BARRIDO	BCDA, 016A
NC BORRA SINCRONO BARRIDO	BCDD, 0170
NC NUEVO SINCRONO RAPIDO	BCE0, 0176
NC AÑADE SINCRONO RAPIDO	BCE3, 017D
NC BORRA SINCRONO RAPIDO	BCE6, 0183
NC AÑADE SINCRONO LENTO	BCE9, 01B3
NC BORRA SINCRONO LENTO	BCEC, 015C
NC INICIALIZA SUCESO	BCEF, 01D2
NC SUCESO	BCF2, 01E2
NC REINICIALIZA SINCRONOS	BCF5, 0228
NC BORRA SUCESO	BCF8, 0285
NC SIGUIENTE SUCESO	BCFB, 0256
NC EJECUTA SUCESO	BCFE, 021A
NC SUCESO EJECUTADO	BD01, 0277
NC INHIBE SUCESOS	BD04, 0295
NC DESINHIBE SUCESOS	BD07, 029B
NC IGNORA SUCESO	BD0A, 028E
NC TIEMPO	BD0D, 0099
NC PONE TIEMPO	BD10, 00A3
MC CARGA Y EJECUTA	BD13, 05DC
MC EJECUTA PROGRAMA	BD16, 060B
MC ESPERA BARRIDO	BD19, 07BA

MC MODO PANTALLA	BD1C, 0776
MC OFFSET PANTALLA	BD1F, 077C
MC BORRA TINTAS	BD22, 0786
MC ASIGNA TINTAS	BD25, 0799
MC INICIA IMPRESORA	BD28, 07E6
MC IMPRIME CARACTER	BD2B, 07F2
MC IMPRESORA OCUPADA	BD2E, 081B
MC ENVIA CARACTER	BD31, 0807
MC REGISTRO SONIDO	BD34, 0826
RESTAURA BLOQUE SALTOS	BD37, 0888
TXT CURSOR ACTIVADO	BDCD, 1263
TXT CURSOR ELIMINADO	BDD0 1263
TXT ESCR CARACTER	BDD3, 134A
TXT LEE CAR	BDD6, 13C0
TXT CODIGO CARACTER	BDD9, 140C
GRA PUNTO	BDDC, 1816
GRA TEXT PUNTO	BDDF, 182A
GRA LINEA	BDE2, 183C
PNT LEE PIXEL	BDE5, 0C82
PNT ESCRIBE PIXEL	BDE8, 0C68
PNT LIMPIA MODO	BDEB, 0AF7
TCL EXAMINA ESC	BDEE, 1C90
MC ESPERA IMPRESORA	BDF1, 07F8

Apéndice C
Mapa de memoria



Apéndice D

Rutinas específicas del 6128

La versión del sistema operativo del CPC6128 no difiere mucho, como cabría esperar, de la primitiva del 464. Si has leído atentamente la primera parte del libro, podrás intuir que añadir 64 Kbytes de memoria RAM no es una tarea inabordable, sino simplemente un pequeño problema de conmutación de bancos de memoria. El sistema operativo incorpora 12 nuevas entradas para el *firmware*. Algunas de ellas ya existían en la versión primitiva, pero no eran accesibles al usuario por el simple uso del CALL. Otras son realmente nuevas y participan en funciones que potencian aún más la habilidad del aparato.

Los bloques de trabajo de las distintas partes del sistema operativo presentan una nueva disposición. Concretamente se han invertido, estando al final el bloque de sucesos, que anteriormente era el primero que encontrábamos. Podría pensarse que la unidad de disco integrada complica enormemente el mapa de memoria, pero no es así en absoluto. La unidad de discos tiene una ROM de 16 Kbytes dedicada exclusivamente a su control. Esta ROM está solapada con la ROM superior y la del intérprete de BASIC. Cuando trabajamos en modo BASIC, la ROM del intérprete está activándose continuamente y alternando con la memoria de pantalla. De igual forma, cuando necesitamos utilizar la unidad de disco, su ROM se activará, alternándose con la de pantalla si es necesario.

La única modificación que hay en el mapa de memoria es la captación de un bloque de la memoria, que sirva como espacio de trabajo de la ROM de disco.

Es de destacar la revisión que se ha hecho del sistema operativo. La inclusión de nuevas subrutinas ha obligado a reducir el espacio que

ocupaban las anteriores. El resultado es un sistema razonablemente más rápido, pero también más complejo. Lo mismo cabe decir de la ROM del intérprete de BASIC en su versión 3.0. La incorporación de las nuevas órdenes e instrucciones ha obligado a perfeccionar en rapidez y efectividad. Y, sobre todo, a reducir en espacio el código máquina de dicha ROM.

El mapa de entrada/salida conserva la configuración expuesta en el capítulo 1. Los contenidos aquí presentados del registro BC combinados con las órdenes IN(C),C o OUT(C),C nos permiten seleccionar los siguientes destinos:

7FXX Matriz de puentes de video (sólo salidas)
BXXX Controlador del tubo de rayos catódicos (TRC), siendo:
- BCXX Salida al registro elegido
- BDXX Salida de datos
- BEXX Registro de estado de entrada
- BFXX Entrada de datos

DFXX Salida de datos de la ROM seleccionada
EFXX Canal de impresora
F4XX Puerto A de la PPI (E/S)
F5XX Puerto B de la PPI (E/S)
F6XX Puerto C de la PPI (E/S)
F7XX Registro de control de la PPI (sólo salidas)
F8FF Reinicialización de los canales de expansión
FBXX Canal para expansiones, siendo:
- FB7X Sistema de disco
- FBBX Una función reservada
- FBDX Un canal de comunicación

Las direcciones que puede emplear el usuario son:

F8E0- F8FE; F9E0-F9FF; FAE0-FAFF; FBE0-FBFF

El proceso de inicialización obedece fielmente a la descripción dada al comienzo del capítulo 3, con la excepción de que implícitamente queda seleccionado el sistema de disco.

Las entradas del tipo RST propias del microprocesador Z-80 están descritas en el capítulo 2 y lógicamente no presentan alteración alguna, ya que son datos que precisa el Z-80 y no deben alterarse.

Acceso a la ROM de disco

La ROM número 07 del sistema contiene 16 Kbytes de memoria con el código máquina necesario para manejar con bastante precisión la unidad de disco y todo lo relacionado con el almacenamiento de datos en

memoria y en las pistas y sectores de los discos. A diferencia de la ROM del intérprete de BASIC, la ROM de disco posee más de una entrada del tipo RSX.

El acceso a esta memoria no tienen ningún misterio. Se acude primero a SELECCIONA ROM con C conteniendo 07; a continuación, se llama a ACTIVA ROM SUPERIOR y, finalmente, saltamos a una de las muchas posibles entradas del sistema. Las direcciones más relevantes son las relacionadas con las órdenes RSX. Siendo éstas:

C006	CPM ROM	Desde donde se produce un salto a C1BC
C009	CPM	Salto a C1B2
C00C	DISC	Salto a CCD1
C00E	DISC.IN	Salto a CCD5
C012	DISC.OUT	Salto a CCE4
C015	TAPE	Salto a CCFD
C018	TAPE.IN	Salto a CD01
C01B	TAPE.OUT	Salto a CD18
C01E	A	Salto a CDDA
C021	B	Salto a CDDD
C024	DRIVE	Salto a CDE4
C027	USER	Salto a CDFE
C02A	DIR	Salto a D42E
C02D	ERA	Salto a D48A
C030	REN	Salto a D4C4

El resto de las entradas funcionará únicamente desde AMSDOS.

Al final del apartado dedicado a las direcciones de entrada al sistema operativo, se describen las entradas de acceso al monitor de la unidad de disco. Sería necesario un libro entero para describir las funciones de cada una de las rutinas de la ROM de disco. Además, debido a la tortuosidad de algunas de las rutinas, no es posible resumir en pocas palabras su labor.

Es de destacar, sin embargo, la doble finalidad de las entradas del bloque de saltos que van desde la dirección BC77 hasta la BC9D. Estas entradas se distinguen, primero, porque no utilizan el código &CF propio de SALTO INF, sino que corresponden a saltos del tipo LLAMADA (RST&18; código &DF), desde donde efectúa un salto a la ROM de disco a través de los tres bytes característicos situados en (A88B/D) (véase capítulo 2).

Desde la ROM de disco se examina en qué modo de unidad de almacenamiento se encuentra el sistema. Si está en el modo cassette (|TAPE), se pasa el control de la ROM inferior, donde están situadas las rutinas de control del cassette. En cambio, si el sistema se encuentra en modo normal (|DISC), la acción continuará en la ROM de disco, utilizando además un espacio de trabajo en RAM, distinto al del empleado para el cassette.

En la inicialización, la ROM &07 reclama un espacio de trabajo que comprende las áreas A67B-AD33 y BE00-BFFF de la RAM.

Cada disco tiene dos caras utilizables. Cada cara con capacidad de 180K se divide de la siguiente forma, según el formato.

Formato del sistema:

40 pistas por cara, numeradas del 0 al 39
9 sectores por pista, numerados del &41 al &49
2 pistas reservadas (0 y 1)

Formato de sólo datos:

40 pistas por cara (0 a 39)
9 sectores por pista, numerados del &C1 al &C9
No hay pistas reservadas

Formato IBM:

40 pistas por cara (0 a 39)
8 sectores por pista (1 al 8)
1 pista reservada

Para todos los formatos, cada sector tiene capacidad para 512 bytes. El máximo de entradas del directorio es 64.

El monitor de la unidad de disco

Todas estas entradas son indirecciones, por lo que la ROM de disco debe estar previamente activada.

DISC MENSAJES: BE80, CA72

Activa o desactiva los mensajes de error para disco.

A la entrada, A contiene &00 para activar los mensajes y &FF para desactivarlos.

A la salida, A contiene el estado anterior.

INICIALIZA DISCO: BE83, C60D

Inicializa los diferentes parámetros relativos a la unidad de disco (UD).

A la entrada, HL contiene la dirección de un bloque de 9 parámetros. La interpretación de bloque es la siguiente:

Bytes 0,1	Tiempo de arranque del motor de la UD en unidades de 1/50 segundos. (50)
Bytes 2,3	Tiempo de parada del motor en unidades de 1/50 segundos. (250)
Byte 4	Tiempo de escritura en unidades de 10 microsegundos. (175)
Byte 5	Tiempo de asentamiento de la cabeza lectora en unidades de 1 milisegundo. (15)
Byte 6	Tiempo entre paso y paso en unidades de 1 milisegundo. (12)
Byte 7	Retardo de escritura de la cabeza lectora. (1)
Byte 8	Retardo de carga de la cabeza lectora. (1)

SELECCIONA FORMATO: BE86, C581

Selecciona uno de los formatos dispuestos para AMSTRAD.

A la entrada, A contiene el número del primer sector del formato elegido. Siendo: &41, formato del sistema; &C1, formato de sólo datos; &01, formato IBM. Además, E contiene 0 si se refiere a la unidad A y E=1 si se refiere a la unidad de disco B.

LEE SECTOR: BE89, C666

Lee un sector físico del disco.

Requisitos de entrada: HL contiene la dirección de un *buffer* en memoria, para el sector; E el número de unidad de disco (0 para la A y para la B); D el número de pista, y C el número de sector.

A la salida, el acarreo estará a uno si se efectuó la lectura correctamente y HL contiene la dirección del *buffer* donde está almacenado el sector. Si hubo error, acarreo a cero.

ESCRIBE SECTOR: BE8C, C64E

Escribe un sector físico del disco.

Requisitos de entrada: HL contiene la dirección del área de memoria donde están almacenados los datos a escribir; E el número de la unidad de disco; D el número de pista, y C el número de sector.

A la salida, el acarreo a uno, si se efectuó la escritura correctamente, y HL contendrá la dirección del *buffer* de datos. Si hubo error, acarreo a cero; A contiene byte de error, y HL, la dirección del *buffer* de errores.

FORMATEA PISTA: BE8F, C652

Formatea una pista según el formato actual.

A la entrada, E contiene el número de unidad de disco; D el número de pista, y HL la dirección de un *buffer* de información de cabeceras. Este *buffer* se interpreta así:

— Entrada al sector para el primer sector:

byte 0: número de pista
byte 1: número de cabecera
byte 2: número de sector
byte 3: $\log_2$ (n.º de bytes del sector) -7

— Entrada al sector para el segundo sector:

byte 0: número de pista
byte 1: ....

— — — — — — — — — — —

— Entrada al sector para el último sector: igual.

A la salida, acarreo a uno si se formateó correctamente; acarreo a cero si hubo error.

LOCALIZA PISTA: BE92, C763

Mueve la cabeza lectora a la pista especificada.

A la entrada, E contiene el número de unidad de disco, y D, el número de pista.

A la salida, acarreo a uno si todo fue bien; acarreo a cero si hubo algún fallo.

OBTIENE ESTADO: BE95, C630

Devuelve el estado de la unidad de disco especificada.

A la entrada, A contiene el número de UD. (0 para la A, 1 para la B).

A la salida, acarreo a uno si no hubo error; A contiene el byte de estado. La interpretación de este byte es:

Bit 0	Unidad seleccionada
Bits 1,2	Siempre a cero
Bit 3	Indefinido
Bit 4	Línea de la pista cero a uno
Bit 5	Línea *ready* a uno. Preparada
Bit 6	Línea de protección contra escritura a uno
Bit 7	Indefinido

FIJA REINTENTOS: BE98, C603

Determina el número de veces que se vuelve a intentar una operación en caso de error.

A la entrada, A contiene el nuevo valor.

A la salida, A contiene el valor anterior.

LLAMADA FIRMWARE: BE9B, C168

Llama a a una rutina del *firmware*. No es recomendable su utilización.

A la entrada, todos los registros y *flags* como requiera la rutina llamada; IX contiene la dirección de la rutina.

A la salida, todos los registros se devolverán tal y como la rutina llamada establezca.

GUARDA REGISTROS: BE9E, C0DB:

Indica si se deben guardar los registros alternativos y el registro IY.

A la entrada, A contiene &00 para guardarlos en la pila, y &FF para no guardarlos.

A la salida, A contiene el valor anterior.

INICIA SIO: BEA1, C389

Inicializa los canales del interfaz serie (SIO).

A la entrada, HL contiene la dirección de un bloque de parámetros. Este se interpreta así:

Byte 0	Registro 4 del canal A de la SIO
Byte 1	Registro 5 del canal A
Byte 2	Registro 3 del canal A
Byte 3	Registro 4 del canal B
Byte 4	Registro 5 del canal B
Byte 5	Registro 3 del canal B
Bytes 6, 7	Temporizador 0 del CI8253
Bytes 8, 9	Temporizador 1 del CI8253
Bytes 10, 11	Temporizador 2 del CI8253

INICIA BUFFER ORDENES: BEA4, C301

Inicializa el *buffer* que contiene las órdenes de comienzo. Los caracteres se introducirán desde el teclado.

A la entrada, A=&00 para vaciar el *buffer* al pulsar una tecla; A=&FF para añadir al *buffer* la tecla pulsada; HL contiene la dirección del *buffer*.

ESTADO ENTRADA D0: BEA7, C3DB

Examina si hay algún carácter disponible en el dispositivo 0 de E/S. A la salida, A contiene &FF si hay carácter, o &00 si no lo hay. El dispositivo 0 tomado por defecto es el canal A de la SIO.

E/S ENTRADA D0: BEAA, C3F7

Toma un carácter del dispositivo 0 de E/S. Si no hay ninguno disponible, esperará hasta que aparezca uno.

A la salida, A contiene el carácter de entrada.

ESTADO SALIDA D0: BEAD, C435

Examina si hay algún carácter preparado para salir en el dispositivo 0 de I/O.

A la salida, A contiene &FF si está preparado, A=&00, si el dispositivo está ocupado.

E/S SALIDA D0: BEB0, C445

Envía un carácter al dispositivo 0 de E/S. Si el dispositivo está ocupado, esperará hasta que dé la señal de listo.

A la entrada, C contiene el carácter a enviar.

ESTADO ENTRADA D1: BEB3, C363

E/S ENTRADA D1: BEB6, C3FF

ESTADO SALIDA D1: BEB9, C43A

E/S SALIDA D1: BEBC, C44B

Estas cuatro rutinas son idénticas a las cuatro anteriores, pero refiriéndose al dispositivo 1 de entrada/salida. Se toma por defecto el canal B de la SIO.

Copia del encabezamiento de ficheros de salida (128 bytes)

Byte 0	Número de usuario (0 a &FF)
Bytes 1-8	Nombre de fichero
Bytes 9-11	Identificador de tipo
Bytes 12-17	Puestos a cero
Byte 18	Tipo de fichero
Bytes 19-20	Longitud del *buffer*
Bytes 21-22	Dirección de lectura de datos
Byte 23	*Flag* de primer bloque. Puesto a &FF
Bytes 24-25	Longitud de los datos
Bytes 26-27	Dirección de comienzo de ejecución
Bytes 28-63	Indefinidos

Bytes 64-66	Longitud del fichero en bytes
Bytes 67-68	Resultado de la suma de los bytes 0 al 66 *(Checksum)*
Resto	Indefinido

Copia del encabezamiento de ficheros de entrada

Posición:	A756	Número de usuario
	A757/5E	Nombre de fichero
	A75F/61	Identificador de tipo
	A762/67	Puestos a cero
	A768	Tipo de fichero
	A769/6A	Dirección del *buffer*
	A76B/6C	Dirección de comienzo
	A76D	*Flag* de primer bloque
	A76E/6F	Longitud total
	Resto	Indefinido

Rutinas del *firmware* en el 6128 (bloque de saltos)

A continuación se indican las direcciones de cada una de las entradas contenidas en la parte medida de la RAM y que de una forma u otra acceden al sistema operativo. La primera dirección hace referencia a la dirección de acceso contenida en la RAM y la segunda dirección indica el comienzo de la rutina asociada a dicha entrada y situada en la ROM inferior, o superior para el sistema de disco.

```
B900.BA5F          ACTIVA ROM SUPERIOR
B903.BA66          INHIBE ROM SUPERIOR
B906.BA51          ACTIVA ROM INFERIOR
B90C.BA70          RESTAURA ROM
B90F.BA79          SELECCIONA ROM
B912.BA9D          ROM ACTUAL
B915.BA7E          TIPO ROM
B918.BA87          ROM ANTERIOR
B91B.BAA1          LDIR
B91E.BAA7          LDDR
B921               NC PRIORIDAD

000B.B984          INF SALTOHL
0008.B98A          INF SALTO
```

001B.B9B9	SALTO HL
0023.B9C1	LLAMADA HL
0018.B9C7	LLAMADA
0013.BA17	SALTO HL LATERAL
0010.BA1D	LLAMADA LATERAL
0028.BA35	SALTO ROM INFERIOR
0020.BAC6	LAM RAM
BB00.1B5C	TCL INICIALIZACION
BB03.1B98	TCL REINICIALIZACION
BB06.1BBF	TCL LEE CARACTER
BB09.1BC5	TCL ESPERA CARACTER
BB0C.1BFA	TCL REGRESA CARACTER
BB0F.1C46	TCL PONE EXPANSION
BB12.1CB3	TCL TOMA EXPANSION
BB15.1C04	TCL ASIGNA BUFFER
BB18.1CDB	TCL ESPERA TECLA
BB1B.1CE1	TCL LEE TECLA
BB1E.1E45	TCL EXAMINA TECLA
BB21.1D38	TCL TOMA ESTADO
BB24.1DE5	TCL ESTADO JOYSTICK
BB27.1ED8	TCL PONE TRASLADO
BB2A.1EC4	TCL TOMA TRASLADO
BB2D.1EDD	TCL TRASLADO CON SHIFT
BB30.1EC9	TCL OBTIENE SHIFT
BB33.1EE2	TCL TRASLADO CON CONTROL
BB36.1ECE	TCL OBTIENE CONTROL
BB39.1E34	TCL IMPONE REPETICION
BB3C.1E2F	TCL CONOCE REPETICION
BB3F.1DF6	TCL IMPONE RETARDO
BB42.1DF2	TCL CONOCE RETARDO
BB45.1DFA	TCL PERMITE RUPTURAS
BB48.1E0B	TCL INHIBE RUPTURAS
BB4B.1E19	TCL GENERA RUPTURA
BB4E.1074	TXT INICIALIZA
BB51.1084	TXT REINICIALIZA
BB54.1459	TXT ACTIVA VDU
BB57.1452	TXT DESACTIVA VDU
BB5A.13FE	TXT SALIDA
BB5D.1335	TXT ESCRIBE CARACTER
BB60.13AC	TXT LEE CARACTER
BB63.13A8	TXT ADMISION DE GRAFICOS
BB66.1208	TXT ACTIVA VENTANA
BB69.1252	TXT EXAMINA VENTANA

BB6C.154F	TXT LIMPIA VENTANA
BB6F.115A	TXT FIJA COLUMNA
BB72.1165	TXT FIJA FILA
BB75.1170	TXT FIJA CURSOR
BB78.117C	TXT EXAMINA CURSOR
BB7B.1286	TXT ACTIVA CURSOR USUARIO
BB7E.1297	TXT INHIBE CURSOR USUARIO
BB81.1276	TXT ACTIVA CURSOR SISTEMA
BB84.127E	TXT INHIBE CURSOR SISTEMA
BB87.11CA	TXT POSICION VALIDA
BB8A.1265	TXT PONE CURSOR
BB8D.1265	TXT QUITA CURSOR
BB90.12A6	TXT COLOR PLUMA
BB93.12BA	TXT EXAMINA PLUMA
BB96.12AB	TXT COLOR PAPEL
BB99.12C0	TXT EXAMINA PAPEL
BB9C.12C6	TXT INVIERTE COLORES
BB9F.137B	TXT FIJA MODO
BBA2.1388	TXT EXAMINA MODO
BBA5.12D4	TXT EXAMINA MATRIZ
BBA8.12F2	TXT IMPONE MATRIZ
BBAB.12FE	TXT FIJA TABLA MATRICES
BBAE.132B	TXT EXAMINA TABLA MATRICES
BBB1.14D4	TXT EXAMINA TABLA CONTROL
BBB4.10E4	TXT SELECCIONA CAUCE
BBB7.1103	TXT CANJEA CAUCES
BBBA.15A8	GRA INICIALIZA
BBBD.15D7	GRA REINICIALIZA
BBC0.15FE	GRA MOV ABSOLUTO
BBC3.15FB	GRA MOV RELATIVO
BBC6.1606	GRA POSICION CURSOR
BBC9.160E	GRA FIJA ORIGEN
BBCC.161C	GRA OBTIENE ORIGEN
BBCF.16A5	GRA ANCHURA VENTANA
BBD2.16EA	GRA ALTURA VENTANA
BBD5.1717	GRA OBTIENE ANCHURA
BBD8.172D	GRA OBTIENE ALTURA
BBDB.1736	GRA LIMPIA VENTANA
BBDE.1767	GRA COLOR PLUMA
BBE1.1775	GRA EXAMINA PLUMA
BBE4.176E	GRA COLOR PAPEL
BBE7.177A	GRA EXAMINA PAPEL
BBEA.1783	GRA PUNTO ABSOLUTO

BBED.1780	GRA PUNTO RELATIVO
BBF0.1797	GRA TEST PUNTO ABS
BBF3.1794	GRA TEST PUNTO REL
BBF6.17A9	GRA LINEA ABSOLUTA
BBF9.17A6	GRA LINEA RELATIVA
BBFC.1940	GRA ESCRIBE CARACTER
BBFF.0ABC	PNT INICIALIZA
BC02.0AD0	PNT REINICIALIZA
BC05.0B37	PNT IMPONE OFFSET
BC08.0B3C	PNT IMPONE BASE RAM
BC0B.0B56	PNT LOCALIZA RAM PANTALLA
BC0E.0AE9	PNT FIJA MODO
BB11.0B0C	PNT EXAMINA MODO
BB14.0B17	PNT BORRA PANTALLA
BB17.0B5D	PNT LIMITES PANTALLA
BB1A.0B6A	PNT POSICION COORDENADAS
BB1D.0BAF	PNT COORDENADA PUNTOS
BC20.0C05	PNT SIGUIENTE BYTE
BC23.0C11	PNT BYTE ANTERIOR
BC26.0C1F	PNT SIGUIENTE LINEA
BC29.0C39	PNT LINEA ANTERIOR
BC2C.0C8E	PNT CODIFICA TINTA
BC2F.0CA7	PNT DECODIFICA TINTA
BC32.0CF2	PNT IMPONE TINTA
BC35.0D1A	PNT OBTIENE TINTA
BC38.0CF7	PNT IMPONE BORDE
BC3B.0D1F	PNT OBTIENE BORDE
BC3E.0CEA	PNT FIJA FLASH
BC41.0CEE	PNT EXAMINA FLASH
BC44.0DB9	PNT RELLENA CAJA
BC47.0DBD	PNT RELLENA BYTE
BC4A.0DE5	PNT INVIERTE CARACTER
BC4D.0E00	PNT DESPLAZA PANTALLA
BC50.0E44	PNT DESPLAZA AREA
BC53.0EF9	PNT EXPANDE
BC56.0F2A	PNT COMPRIME
BC59.0C55	PNT MODO GRAFICOS
BC5C.0C74	PNT PIXEL
BC5F.0F93	PNT HORIZONTAL
BC62.0F9B	PNT VERTICAL
BC65.24BC	CAS INICIALIZA
BC68.24CE	CAS VELOCIDAD
BC6B.24E1	CAS MENSAJES

BC6E,2BBB	CAS ARRANCA MOTOR
BC71,2BBF	CAS PARA MOTOR
BC74,2BC1	CAS RESTAURA MOTOR
BC77,24E5	DCAS ABRE ENTRADA
BC7A,2550	DCAS CIERRA ENTRADA
BC7D,257F	DCAS ABANDONA ENTRADA
BC80,25A0	DCAS LEE CARACTER
BC83,2618	DCAS ENTRADA DIRECTA
BC86,2607	DCAS RETORNO
BC89,2603	DCAS EXAMINA FINAL
BC8C,24FE	DCAS ABRE SALIDA
BC8F,256D	DCAS CIERRA SALIDA
BC92,2599	DCAS ABANDONA SALIDA
BC95,25C6	DCAS SALIDA CARACTER
BC98,2653	DCAS SALIDA DIRECTA
BC9B,2692	DCAS CATALOGA
BC9E,29AF	CAS ESCRIBE
BCA1,2946	CAS LEE
BCA4,29C1	CAS COMPARA
BCA7,1FE9	REINICIALIZA SONIDO
BCAA,2114	COLA SONIDO
BCAD,21CE	EXAMINA COLA
BCB0,21EB	FORMA SUCESO SONIDO
BCB3,21AC	EJECUTA SONIDO
BCB6,2050	PARA SONIDO
BCB9,206B	CONTINUA SONIDO
BCBC,2495	ENVOLVENTE SONIDO
BCBF,249A	ENVOLVENTE TONO
BCC2,24A6	DIRECCION ENVOLVENTE A
BCC5,24AB	DIRECCION ENVOLVENTE T
BCC8,005C	NC REINICIALIZA NUCLEO
BCCB,0326	NC ROM SECUNDARIA
BCCE,0330	NC INIC SECUNDARIA
BCD1,02A0	NC INTRODUCE RSX
BCD4,02B1	NC BUSCA ORDEN
BCD7,0163	NC NUEVO SINCRONO BARRIDO
BCDA,016A	NC ANADE SINCRONO BARRIDO
BCDD,0170	NC BORRA SINCRONO BARRIDO
BCE0,0176	NC NUEVO SINCRONO RAPIDO
BCE3,017D	NC ANADE SINCRONO RAPIDO
BCE6,0183	NC BORRA SINCRONO RAPIDO
BCE9,01B3	NC ANADE SINCRONO LENTO
BCEC,01C5	NC BORRA SINCRONO LENTO

BCEF.01D2	NC INICIALIZA SUCESO
BCF2.01E2	NC SUCESO
BCF5.0227	NC REINICIALIZA SINCRONOS
BCF8.0284	NC BORRA SUCESO
BCFB.0255	NC SIGUIENTE SUCESO
BCFE.0219	NC EJECUTA SUCESO
BC01.0276	NC SUCESO EJECUTADO
BC04.0294	NC INHIBE SUCESOS
BC07.029A	NC DESINHIBE SUCESOS
BC0A.028D	NC IGNORA SUCESOS
BC0D.0099	NC TIEMPO
BD10.00A3	NC PONE TIEMPO
BD13.05ED	MC CARGA Y EJECUTA
BD16.061C	MC EJECUTA PROGRAMA
BD19.07B4	MC ESPERA BARRIDO
BD1C.0776	MC MODO PANTALLA
BD1F.07C0	MC OFFSET PANTALLA
BD22.0786	MC BORRA TINTAS
BD25.078C	MC ASIGNA TINTAS
BD28.07E0	MC INICIA IMPRESORA
BD2B.081B	MC IMPRIME CARACTER
BD2E.0858	MC IMPRESORA OCUPADA
BD31.0844	MC ENVIA CARACTER
BD34.0863	MC REGISTRO SONIDO
BD37.08BD	RESTAURA BLOQUE SALTOS
BD3A.1D3C	TCL PONE ESTADO
BD3D.1BFE	TCL TECLADO LIBRE
BD40.1460	TXT CURSOR
BD43.15EC	GRA MODO ABSOLUTO
BD46.19D5	GRA OBTIENE MODO GRAFICOS
BD49.17B0	
BD4C.17AC	
BD55.0B45	PNT BASE Y OFFSET
BDCD.125F	TXT CURSOR ACTIVADO
BDD0.125F	TXT CURSOR ELIMINADO
BDD3.134B	TXT ESCRIBE CARACTER
BDD6.13BE	TXT LEE CARACTER
BDD9.140A	TXT CODIGO CARACTER

```
BDDC,1786          GRA PUNTO
BDDF,179A          GRA TEST PUNTO
BDE2,17B4          GRA LINEA

BDE5,0C8A          PNT LEE PIXEL
BDE8,0C71          PNT ESCRIBE PIXEL
BDEB,0B17          PNT LIMPIA MODO

BDEE,1DB8          TCL EXAMINA ESC
BDF1,0835          MC ESPERA IMPRESORA

BE80,CA72          DICS MENSAJES
BE83,C60D          INICIALIZA DISCO
BE86,C581          SELECCIONA FORMATO
BE89,C666          LEE SECTOR
BE8C,C64E          ESCRIBE SECTOR
BE8F,C652          FORMATEA PISTA
BE92,C763          LOCALIZA PISTA
BE95,C630          OBTIENE ESTADO
BE98,C603          FIJA REINTENTOS
BE9B,C168          LLAMADA FIRMWARE
BE9E,C0DB          GUARDA REGISTROS
BEA1,C389          INICIA SIO
BEA4,C301          INICIA BUFFER ORDENES
BEA7,C3DB          ESTADO ENTRADA D0
BEAA,C3F7          E/S ENTRADA D0
BEAD,C435          ESTADO SALIDA D0
BEB0,C445          E/S SALIDA D0
BEB3,C3E3          ESTADO ENTRADA D1
BEB6,C3FF          E/S ENTRADA D1
BEB9,C43A          ESTADO SALIDA D1
BEBC,C44B          E/S SALIDA D1
```

Bloque de saltos para el núcleo

Este grupo de entradas permite activar o desactivar cualquier ROM incluida en el sistema para poder acceder a su contenido. Existen doce entradas localizadas entre las posiciones B900 y B921.

ACTIVA ROM SUPERIOR: B900, BA5F

La ROM superior seleccionada previamente por SELECCIONA ROM se activa. No hay condiciones de entrada. Al regreso, A contiene el byte de estado previo de la ROM.

INHIBE ROM SUPERIOR: B903, BA66

Se inhibe la ROM previamente seleccionada. Sin condiciones de entrada. Al regreso, A contiene el estado anterior de ROM.

ACTIVA ROM INFERIOR: B906, BA51

El mismo proceso que para ACTIVA ROM SUPERIOR, aunque no hace falta seleccionarla.

INHIBE ROM INFERIOR: B909, BA58

La ROM inferior es inhibida. A contiene, al regreso, el estado previo de ROM.

En estas cuatro rutinas, el estado de ROM contenido en A deberá transferirse a RESTAURA ROM para recuperar la ROM que había antes de que dichas rutinas fueran llamadas.

RESTAURA ROM: B90C, BA70

A debe contener el byte de estado de la ROM. Este estado pasa entonces a ser el actual. No hay condiciones de salida. AF, alterados.

SELECCIONA ROM: B90F, BA79

A su entrada, C contiene el número de ROM que se quiere seleccionar. A su salida, C contiene el número de la ROM anterior, y B el byte de estado de la misma.

ROM ACTUAL: B912, BA7D

Sin condiciones de entrada. A su salida, A contiene el número de la ROM actual. Este dato lo obtiene del contenido de la dirección B8D6.

TIPO ROM: B915, BA7E

Al entrar, C contiene la dirección de la ROM a comprobar, la cual debe estar previamente seleccionada (no activada). A su salida, A contiene el tipo de ROM; H, el número de versión de la ROM, y L es número de marca. El significado del contenido de A es el siguiente:

0: ROM primaria
1: ROM secundaria o subordinada
2: Extensión de la ROM
&80: Es la propia del sistema

ROM ANTERIOR: B918, BA87

Al comienzo, C contiene el número de ROM requerida, y B, su byte de estado. A su salida, C contiene el estado de la ROM anterior, es decir, la que antes de la llamada era la ROM actual. La ROM seleccionada por esta rutina es la que había antes de acudir a SELECCIONA ROM.

LDIR: B91B, BAA1

LDDR: B91E, BAA7

Efectúan la operación LDIR o LDDR del Z-80, pero con las ROM superior e inferior inhibidas. El estado en que se encontrarán las ROM no afecta con comportamiento de la rutina y, además, dicho estado se mantendrá a la salida. El requisito de entrada es que BC, DE y HL contengan los mismos valores que tendrían en la misma instrucción de Z-80.

NC PRIORIDAD: B921

Véase apartado del núcleo.

Area de extensiones RST

Este bloque ya fue expuesto en el capítulo 2 con todo detalle. Indico aquí las nuevas direcciones asociadas en la RAM.

0008, B98A: INF SALTO
000B, B984: INF SALTO HL
0010, BA1D: LLAMADA LATERAL

0013, BA17:	SALTO HL LATERAL
0018, B9C7:	LLAMADA
001B, B9B9:	SALTO HL
0020, BAC6:	LAM RAM
0023, BAC1:	LLAMADA HL
0028, BA35:	SALTO ROM INFERIOR

El control de la máquina

Es el grupo encargado de dirigir y controlar los periféricos *hardware* del sistema, así como de la inicialización principal.

MC CARGA Y EJECUTA: BD13, 05ED

A su entrada, HL debe contener la dirección de una rutina que se encargará de cargar el programa. Esta rutina deberá estar en RAM en el margen 4000-B000 o en la misma ROM. Si la carga se efectuó sin fallos (acarreo a uno), entonces HL contendrá la dirección del punto de entrada al programa. Si HL = 0000, entonces se pasará el control al intérprete de BASIC en la dirección C000. Si el programa cargado termina de ejecutarse, se llevará a cabo una completa reinicialización del sistema. CALL 0.

MC EJECUTA PROGRAMA: BD16, 061C

A su entrada, HL contiene la dirección de comienzo, y C el número de ROM a emplear. Al comienzo de esta rutina y de la anterior se produce una inicialización completa del sistena y después se ejecuta el programa.

Rutinas de acceso a la impresora

MC INICIA IMPRESORA: BD28, 07E0

Reinicializa la indirección BDF1. Con ello se consigue que el gobierno de la impresora corra a cargo del propio sistema operativo y no de algún programa introducido por el usuario.

MC IMPRIME CARACTER: BD2B, 081B

Al entrar, A contiene el carácter a enviar a la impresora. Se llama a MC ESPERA IMPRESORA. Al terminar la rutina, se puede comprobar si el carácter fue enviado, observando el acarreo. Si se envió, está a uno.

MC IMPRESORA OCUPADA: BD2E, 0858

Comprueba si el puerto Centronics está ocupado. Si al regreso el acarreo está a uno, el puerto estará ocupado. Si no lo está, C = 0.

MC ENVIA CARACTER: BD31, 0844

Envía un carácter a la impresora sin preguntar antes si ésta estaba ocupada o no. A su entrada, A contiene el carácter a enviar.

MC ESPERA IMPRESORA: BDF1, 0835

Es una indirección. A su entrada, A contiene el carácter a enviar a la impresora. La rutina esperará hasta que ésta esté desocupada, pero, si se retrasa mucho, la rutina regresa y no se envía el carácter. A la salida, el acarreo estará a uno si se envió el carácter.

Rutinas referentes a la pantalla

MC BORRA TINTAS: BD22, 0786

A su entrada, DE contiene la dirección de un vector de tintas. Este vector es de la forma:

Byte 0: Color del borde
Byte 1: Color para todas las tintas

Con esta rutina, todas las tintas son asignadas a un mismo color.

MC ASIGNA TINTAS: BD25, 078C

La rutina asigna un color a cada una de las tintas y al borde. A la entrada, DE debe contener la dirección de un vector de tintas de la forma;

Byte 0: Color del borde
Byte 1: Color de la tinta 0
Byte 2: Color de la tinta 1

Byte 16: Color de la tinta 15

MC ESPERA BARRIDO: BD19, 07B4

Retrasa toda acción en la pantalla hasta que se produzca un nuevo barrido del haz de rayos catódicos. No hay requisitos de entrada.

MC OFFSET PANTALLA: BD1F, 07C0

Al entrar, A contiene el byte superior de la base requerida, y HL contiene el *offset* o el desplazamiento.

MC MODO PANTALLA: BD1C, 0776

A su entrada, A contiene el modo requerido. La rutina comprobará la validez de este número. Este es el cambio de modo *hardware*, que siempre debe ser precedido de un cambio de modo *software* (véase PNT FIJA MODO).

MC REGISTRO SONIDO: BD34, 0863

Al entrar, A contiene el número de registro, y C, el dato a introducir en el registro. El registro es uno de los del generador programable de sonido. La descripción de este registro a nivel profundo está en los catálogos de general instruments. Una descripción más simplificada pero igualmente válida la encontrarás en el libro *Rutinas en lenguaje máquina para Amstrad**.

Sería innecesario volver a repetir el comportamiento interno de las rutinas del sistema operativo, y que, excepto las nuevas entradas, el resto

* Pritchard, Joe: *Rutinas en lenguaje máquina para Amstrad*, Anaya Multimedia, 1986.

conserva prácticamente la forma en que estaban organizadas en el sistema operativo del CPC464-6128. Por tanto, cambiando simplemente el comienzo de la rutina en la ROM inferior por la de la siguiente lista, no tendrás problema en identificar cada una de las operaciones tal y como se explican y describen en la primera parte del libro. Aunque no te importe el sistema operativo del 464, es importante que te des cuenta de la similitud de las versiones. No tengas, pues, ningún reparo en hojear detenidamente la descripción de las rutinas del S.O. del Amstrad CPC464-6128.

Hecha esta aclaración, pasaremos a definir la función de cada una de las rutinas del *firmware* del CPC6128.

El núcleo o control de sucesos e interrupciones

NC INICIALIZA SUCESO: BCEF, 01D2

Inicializa un bloque de suceso.

Requisitos de entrada: HL contiene la dirección del bloque; B la clase de suceso; C la dirección de selección (número) de ROM para ese suceso, y DE la dirección de la rutina asociada.

Condiciones de salida: HL contiene la dirección del bloque de suceso más 7.

NC SUCESO: BCF2, 01E2

Atiende un bloque de suceso.

Requisitos de entrada: HL contiene la dirección del bloque de suceso. No hay condiciones de salida.

NC REINICIALIZA SINCRONOS: BCF5, 0227

Borra la secuencia de sucesos síncronos.

No hay requisitos de entrada ni condiciones de salida.

NC BORRA SUCESO: BCF8, 0284

Borra un suceso síncrono de la secuencia de sucesos.

A su entrada, HL debe contener la dirección del bloque de suceso.

NC SIGUIENTE SUCESO: BCFB, 0255

Indica cuál va a ser el siguiente suceso que tendrá lugar. No hay requisitos de entrada. Condiciones de salida: si hay algún suceso que procesar, el acarreo estará a uno; HL contiene la dirección del bloque del suceso, y A la prioridad del último suceso procesado. Si no hay sucesos que procesar, el acarreo estará a cero.

NC NUEVO SINCRONO BARRIDO: BCD7, 0163

Inicializa y añade un bloque en la lista de síncronos asociados al barrido de pantalla.

Requisitos de entrada: HL contiene la dirección del bloque; B la clase de suceso; C el número de ROM de la rutina del suceso, y DE la dirección de la rutina asociada al suceso. No hay datos útiles a la salida.

NC AÑADE SINCRONO BARRIDO: BCDA, 016A

Añade un bloque a la lista de este tipo de sucesos.

Requisitos de entrada: HL contiene la dirección del bloque. Sin condiciones de salida.

BORRA SINCRONO BARRIDO: BCDD, 0170

Elimina un bloque de la lista de este tipo de sucesos.

Requisitos de entrada: HL contiene la dirección del bloque.

NC NUEVO SINCRONO RAPIDO: BCE0, 0176

NC AÑADE SINCRONO RAPIDO: BCE3, 017D

NC BORRA SINCRONO RAPIDO: BCE6, 0183

Estas tres rutinas ejercen la misma función que las anteriores, pero refiriéndose en este caso a los sucesos de temporización rápida. Mismos requisitos de entrada que las anteriores.

NC AÑADE SINCRONO LENTO: BCE9, 01B3

Pone un bloque en la lista de sucesos síncronos lentos.
Requisitos de entrada: HL contiene la dirección del bloque; DE el valor inicial del byte "cuenta", y BC el valor de recarga. Este bloque de 13 bytes debe estar almacenado en los 32 Kbytes centrales de la RAM.

NC BORRA SINCRONO LENTO: BCEC, 01C5

Elimina un bloque de la lista de síncronos lentos.
Requisitos de entrada: HL contiene la dirección del bloque.
Condiciones de salida: Si el bloque se encontró en la lista, el acarreo estará a uno, y DE contiene la cuenta remanente. Si no se encontró, el acarreo estará a cero.

NC IGNORA SUCESO: BD0A, 028D

Impide la ejecución de un suceso. A la entrada, HL contiene la dirección del bloque.

NC EJECUTA SUCESO: BCFE, 0219

Ejecuta la rutina asociada a un suceso. A la entrada, HL contiene la dirección del bloque del suceso.

NC SUCESO EJECUTADO: BD01, 027C

Termina la ejecución de un suceso.
Requisitos de entrada: A contiene la prioridad del suceso anterior, y HL la dirección del bloque del suceso.

NC PRIORIDAD: B921

Comprueba si hay algún suceso con una prioridad mayor que la definida actualmente. Contenido de (B8C2). Si así es, el acarreo se devuelve a nivel alto.

NC INHIBE SUCESOS: BD04, 0294

Inhibe los sucesos síncronos normales, conservando los clasificados como urgentes. No hay requisitos de entrada.

NC DESINHIBE SUCESOS: BD07, 029A

Habilita los sucesos síncronos normales. No hay requisitos de entrada.

NC TIEMPO: BD09, 0099

La rutina regresa con la cuenta del tiempo transcurrido.

Condiciones de salida: DEHL contienen los cuatro bytes del contador. D, el byte más alto, y L, el menos significativo.

NC PONE TIEMPO: BD10, 00A3

Fija un valor inicial para el contador de tiempo.

Requisitos de entrada: DEHL contendrán los cuatro bytes para el contador. D, el byte más significativo, y L, el menos significativo.

AREA DE DATOS DEL NUCLEO

CPC6128		CPC464
B82D-E	Base de la lista de inatendidos	B100-B101
B82F-B830	Fin de la lista de inatendidos	B102-B103
B831	Byte de *flag*	B104
B832-B833	Contiene el SP (puntero de pila)	B105-B106
B834-B8B3	Pila especial	B107-B186
B8B4-B8B8	Contador de tiempo	B187-B18D
B8B9-B8BA	Base de lista de sucesos asociados barridos	B18C-B18D
B8BB-B8BC	Base de lista de temporización rápida	B18E-B18F
B8BD-B8BE	Base de lista de temporización lenta	B190-B191
B8BF	Contador de temporización lenta	B192
B8C0-B8C1	Base de lista de síncronos	B193-B194
B8C2	Prioridad actual	B195
B8C3-B8D2	Copia de las palabras-órdenes	B196-B1A5
B8D3-B8D4	Base de la cadena de órdenes	B1A6-B1A7
B8D6	ROM actual	B1A8
B8D7-B8D8	Dirección lejana característica	B1A9-B1AB
B8D9-B8E5	Punteros del área de datos	B1AC-B1B8

COMPRENSION DE PANTALLA

CPC6128		CPC464
B7C3	Modo	B1C8
B7C4-5	Offset	B1C9-B1CA
B7C6	Base (byte alto)	B1CB
B7C7-6	Instrucción de asaltos	B1CC-B1CE
B7CA-D1	Máscaras de puntos	B1CF-B1D6
B7D2	Segundo tiempo de parpadeo	B1D7
B7D3	Primer tiempo de parpadeo	B1D8
B7D4-E4	Tabla de 2 colores	B1D9-B1E9
B7E5-F5	Tabla de 1 color	B1EA-B1FA
B7F6	*Flag* de selección de tabla	B1FB
B7F7	Contador de parpadeos	B1FC
B7F8	Tiempo de cada color	B1FD
B7F9-B801	Bloque de sucesos	B1FE-B206
B802	Bits que forman el punto, negados	B207

El sistema de pantalla

PNT INICIALIZA: BBFF, 0ABF

Inicialización completa del sistema de pantalla; direcciones, variables, tintas y modo de pantalla. No hay requisitos de entrada.

PNT REINICIALIZA: BC02, 0AD0

Reinicializa las indirecciones, las tablas de color, velocidad de parpadeo y el modo de texto. No hay requisitos de entrada.

PNT BORRA PANTALLA: BDEB, 0B17

Pone toda la pantalla con la tinta 0. No hay requisitos de entrada.

Las rutinas llamadas por PNT BORRA PANTALLA corresponden con otras direcciones a la descripción hecha en la página 62. Estas nuevas direcciones son:

0D42, 0D55, llamadas por BORRA PANTALLA
0D61, rutina de sucesos

0D6D, llamada por 0D42 y 0D61
0D81, llamada por 0D55, 0D61 y 0D42

Ninguna de estas rutinas es accesible desde el bloque de saltos.

PNT FIJA MODO: BC0E, 0AE9

Define el modo de pantalla y actualiza convenientemente la VDU de texto y la de gráficos. A la entrada, A contiene el número de modo.

PNT EXAMINA MODO: BC11, 0B0C

Al regreso, el modo actual queda indicado por el estado de los *flags* de cero y de acarreo.

Modo 0: C=1 Z=0
Modo 1: C=0 Z=1
Modo 2: C=0 Z=0

PNT IMPONE OFFSET: BC05, 0B37

Impone el desplazamiento del origen de la pantalla.
A la entrada, HL contiene el desplazamiento requerido.

PNT IMPONE BASE: BC08, 0B3C

Define la dirección base (origen) de la pantalla. Puede ser 0000, 4000, 8000, o C000. A su entrada, A contiene el byte alto de dicha dirección.

PNT BASE Y OFFSET: BD55, 0B45

Permite definir simultáneamente la base y el desplazamiento de la pantalla. A su entrada, A contiene el byte más alto de la base y HL el desplazamiento.

PNT LOCALIZA RAM PANTALLA: BC0B, 0B56

Examina la dirección base de pantalla y el desplazamiento actual. A la salida, A contiene el byte alto de la base y HL el desplazamiento.

PNT LIMITES PANTALLA: BC17, 0B5D

Examina el tamaño en caracteres de la pantalla.

A la salida, B contiene la última columna física de la pantalla, y C, la última fila física de pantalla.

PNT POSICIONA COORDENADAS: BC1A, 0BCA

Calcula la dirección de pantalla de la esquina superior izquierda de un carácter.

Requisitos de entrada: H contiene la columna física del carácter; L la fila del carácter.

Condiciones de salida: HL contiene la dirección de la esquina superior izquierda del carácter; B contiene el ancho en bytes del carácter.

PNT COORDENADA PUNTOS: BC1B, 0BAF

Calcula la dirección de pantalla de un *pixel*.

Requisitos de entrada: DE contiene la coordenada X del *pixel;* HL contiene la coordenada Y.

Condiciones de salida: HL contiene la dirección de pantalla del *pixel;* C la máscara del *pixel*, y B el número de *pixels* que caben en un byte, menos uno.

PNT SIGUIENTE BYTE: BC20, 0C05

Calcula la dirección del byte situado a la derecha de una dirección dada.

A la entrada, HL contiene una dirección de pantalla. A la salida, HL contendrá la dirección requerida.

PNT BYTE ANTERIOR: BC23, 0C11

Calcula la dirección del byte situado a la izquierda de una dirección dada.

A la entrada, HL contiene una dirección de pantalla. A la salida, HL contendrá la dirección requerida.

PNT SIGUIENTE LINEA: BC26, 0C1F

Calcula la dirección del byte situado una línea por debajo de una dirección dada.

A la entrada, HL contiene una dirección de pantalla. A la salida, HL contiene la dirección de pantalla requerida.

PNT CODIFICA TINTA: BC2C, 0C8E

Codifica una tinta que rellenará todos los *pixels* de un byte.

A la entrada, A contiene un número de tinta. A la salida, A contiene la tinta codificada.

PNT DECODIFICA TINTA: BC2F, 0CA7

Convierte una tinta decodificada en el número de tinta correspondiente.

A la entrada, A contiene una tinta decodificada. A la salida, A contiene el número de tinta.

PNT IMPONE TINTA: BC32, 0CF2

PNT IMPONE BORDE: BC38, 0CF7

Fija respectivamente los colores de un número de tinta y del borde. Si los dos colores especificados a la entrada son distintos, éstos se alternarán en la pantalla en forma de parpadeo.

Requisitos de entrada: B contiene el primer color, y C el segundo color. Además, en el caso de IMPONE TINTA, A contiene el número de tinta a chequear.

Condiciones de salida para ambas rutinas: B contiene el primer color, y C el segundo color.

PNT OBTIENE TINTA: BC35, 0D1A

PNT OBTIENE BORDE: BC3B, 0D1F

Examinan, respectivamente, los colores relacionados con una tinta dada y con el borde. A la entrada de OBTIENE TINTA, A contiene el número de tinta a comprobar. A la salida y en ambas rutinas, B contiene el primer color y C el segundo color.

PNT FIJA FLASH: BC3E, 0CEA

Impone los períodos de parpadeo, en unidades de 1/50 segundos.
A la entrada, H contiene el período del primer color, y L el período del segundo color.

PNT EXAMINA FLASH: BC41, 0CEE

Examina los períodos de parpadeo.
A la salida, H contiene el período del primer color, y L el período del segundo color.

PNT RELLENA CAJA: BC44, 0DB9

Rellena un área rectangular de la pantalla con una tinta dada.
Requisitos de entrada: A contiene la tinta codificada; H la columna física más a la izquierda; D, la columna más a la derecha; L la fila superior, y E contiene la fila inferior.

PNT RELLENA BYTE: BC47, 0DBD

Rellena un área rectangular de la pantalla con una tinta dada.
Requisitos de entrada: C contiene la tinta codificada; HL la dirección de pantalla de la esquina superior izquierda del rectángulo; D el ancho en bytes, y E la altura del rectángulo en líneas de pantalla.

PNT INVIERTE COLORES: BC4A, 0DE5

Intercambia las tintas de papel y pluma de un carácter dado.
Requisitos de entrada: B y C contienen las tintas codificadas; H contiene la columna física del carácter, y L la fila física.

PNT DESPLAZA PANTALLA: BC4D, 0E00

Desplaza la pantalla hacia arriba o hacia abajo ocho líneas. La fila insertada aparece con el color que se especifique.
A la entrada, B=0 si el desplazamiento es hacia abajo, y distinto de

cero si es hacia arriba; A contiene la tinta codificada de la nueva línea que aparece.

PNT DESPLAZA AREA: BC50, 0E44

Desplaza verticalmente un área de la pantalla ocho líneas.

Requisitos de entrada: B=0 indica desplazamiento hacia abajo; B=0 indica desplazamiento hacia arriba; A contiene la tinta codificada de la línea que aparece; H la columna física de la izquierda del área; D la columna derecha; L la fila superior, y E la fila inferior.

PNT EXPANDE: BC53, 0EF9

Expande la matriz que define un carácter para adaptarla al modo actual de pantalla.

A la entrada, HL contiene la dirección donde está definida la matriz, y DE la dirección de un área de memoria que se empleará para realizar la expansión.

PNT COMPRIME: BC56, 0F2A

Convierte un carácter desde el tamaño que tiene en pantalla al tamaño estándar.

Requisitos de entrada: A contiene la tinta codificada del carácter; H la columna física del carácter; L la fila que ocupa el carácter, y DE la dirección del área donde se formalizará la matriz estándar.

PNT MODO GRAFICOS: BC59, 0C55

Fija el modo de gráficos para la unidad de video (VDU).

A la entrada, A contiene el modo requerido, siendo:

A=0 Modo absoluto o forzado
A=1 Modo XOR
A=2 Modo AND
A=3 Modo OR

PNT PIXEL: BC5C, 0C74

Escribe en la pantalla un *pixel*, ignorando el modo de gráficos actual.

Requisitos de entrada: B contiene la tinta codificada del *pixel;* C la máscara para el *pixel*, y HL la dirección de pantalla.

PNT HORIZONTAL: BC5F, 0F93

Dibuja una línea horizontal.

Requisitos de entrada: A contiene la tinta codificada para la línea; DE la coordenada X de comienzo; BC la coordenada X final, y HL la coordenada Y de la línea.

PNT VERTICAL: BC62, OF9B

Dibuja una línea vertical.

Requisitos de entrada: A contiene la tinta codificada; DE la coordenada X de la línea; HL la coordenada Y de comienzo de la línea, y BC la coordenada Y final.

PNT LEE PIXEL: BDE5, 0C8A

Es una indirección. Lee un punto de la pantalla y decodifica su tinta.

Requisitos de entrada: HL contiene la dirección de pantalla del punto, y C la máscara del punto.

Condiciones de salida: A contiene la tinta decodificada del punto.

PNT ESCRIBE PIXEL: BDE8, 0C71

Activa uno o varios puntos de la pantalla en el modo de gráficos actual.

Requisitos de entrada: HL contiene la dirección de pantalla del *pixel;* C la máscara del *pixel*, y B la tinta codificada en la que escribe el *pixel*.

La unidad de video para texto

TXT INICIALIZA: BB4E, 1074

Inicializa la unidad de video para texto, indirecciones y variables. No hay requisitos.

TXT REINICIALIZA: BB51, 1084

Reinicializa las indirecciones y tabla de control. No hay requisitos.

TXT ACTIVA VDU: BB54, 1459

Permite mostrar caracteres en la pantalla en la ventana actual seleccionada. No hay requisitos.

TXT FIJA COLUMNA: BB6F, 115A

Mueve el cursor de la ventana actual a la columna requerida.
A la entrada, A contiene el número de columna lógica requerida.

TXT FIJA FILA: BB72, 1165

Mueve el cursor de la ventana actual a la fila requerida.
A la entrada, A contiene la fila lógica requerida para el cursor.

TXT EXAMINA CURSOR: BB78, 1176

Devuelve la posición del cursor dentro de la ventana y el número de veces que se ha desplazado la ventana.

A la salida, H contiene la columna lógica del cursor; L la fila lógica, y A la cuenta actual de desplazamientos.

TXT ACTIVA CURSOR USUARIO: BB7B, 1286

Habilita el cursor para el cauce actual. No hay requisitos.

TXT INHIBE CURSOR USUARIO: BB7E, 1297

Deshabilita el cursor para el cauce actual. No hay requsitos.

TXT ACTIVA CURSOR SISTEMA: BB81, 1276

Permite mostrar el cursor en pantalla. No hay requisitos.

TXT INHIBE CURSOR SISTEMA: BB84, 127E

Impide mostrar el cursor en pantalla. Estas dos últimas rutinas son utilizadas únicamente por el sistema operativo.

TXT PONE CURSOR: BB8A, 1265

TXT QUITA CURSOR: BB8D, 1265

Pone o quita un cursor de texto en el cauce actual. Se permite un cursor por cada cauce o ventana. No hay requisitos.

TXT CURSOR: BD40, 1460

Permite imponer la activación o desactivación del cursor.

Si a la entrada $A=0$, el cursor quedará activado, pero si $A \neq 0$, el cursor es desactivado.

TXT CURSOR ACTIVADO: BDCD, 125F

TXT CURSOR ELIMINADO: BDD0, 125F

Son dos indirecciones. Colocan o retiran respectivamente el cursor de la pantalla. Para lo primero, el cursor ha de estar previamente habilitado. No hay requisitos.

TXT POSICION VALIDA: BB87, 11CA

Comprueba si la posición del cursor está dentro de los límites de la ventana.

Requisitos de entrada: H contiene la columna lógica de la posición a comprobar, y L la fila lógica.

Condiciones de salida: H contiene la columna lógica donde se impimiría el carácter, y L la fila lógica. Si escribir el carácter no hace que la ventana se desplace, el acarreo estará a uno. Si la ventana se desplazara hacia arriba, el acarreo estaría a cero, y B contendría &FF. Si la ventana se viera obligada a desplazarse hacia abajo, el acarreo estaría a cero y B contendría &00.

TXT COLOR PLUMA: BB90, 12A6

TXT COLOR PAPEL: BB96, 12AB

Definen respectivamente la tinta de escritura y la del papel para el cauce actual. A su entrada, A debe contener el número de la tinta.

TXT EXAMINA PLUMA: BB93, 12BA

TXT EXAMINA PAPEL: BB99, 12C0

Comprueban respectivamente la tinta de escritura y del papel del cauce actual. A la salida, A contiene el número de la tinta.

TXT INVIERTE COLORES: BB9C, 12C6

Intercambia las tintas de la pluma de texto y del papel para el cauce actual. No hay requisitos.

TXT FIJA MODO: BB9F, 137B

Determina si se escribe en modo opaco o transparente. Sólo funciona con el modo forzado. A la entrada, A=0 si se quiere el modo opaco, y A≠0 si se quiere el modo transparente.

TXT EXAMINA MODO: BBA2, 1388

Comprueba el modo de escritura.

A la salida, A=0 si se está utilizando el modo opaco; A≠Ø indica que se emplea el modo transparente.

TXT ACTIVA VENTANA: BB66, 1208

Determina el tamaño de la ventana de texto actual.

Requisitos de entrada: H contiene la columna física de un borde, y D la columna del otro borde; L contiene la fila de un borde, y E la fila del otro borde.

TXT EXAMINA VENTANA: BB69, 1252

Obtiene el tamaño de la ventana actual.

Condiciones de salida: H contiene la columna más a la izquierda de la ventana; D la columna más a la derecha; L la fila superior de la ventana, y E la fila inferior. Además, si la ventana ocupa toda la pantalla, el acarreo regresará a cero. En caso contrario, regresará a uno.

TXT LIMPIA VENTANA: BB6C, 154F

Borra la ventana de texto del cauce actual y la rellena con la tinta del papel.

No hay requisitos.

TXT SELECCIONA CAUCE: BBB4, 10E4

Selecciona un cauce *(stream)*.

A la entrada, A debe contener el número del cauce a seleccionar.

TXT CANJEA CAUCES: BBB7, 1103

Intercambia dos cauces.

A la entrada, C y B contienen los cauces a canjear.

TXT EXAMINA TABLA MATRICES: BBAE, 132B

Comprueba la dirección de la actual tabla de matrices definidas por el usuario y el primer carácter de la tabla.

Condiciones de salida: si no hay ninguna tabla definida por el usuario, el acarreo estará a cero; si hay alguna tabla definida, el acarreo estará a uno; A contendrá el código del primer carácter de la tabla, y HL la dirección de comienzo de la tabla.

TXT EXAMINA MATRIZ: BBA5, 12D4

Obtiene la dirección del modelo de un carácter.

A la entrada, A contiene el código del carácter que ha de encontrarse.

A la salida, el acarreo estará a uno si la matriz está en la tabla definida por el usuario, pero estará a cero si la matriz pertenece a las definiciones de la ROM baja.

En cualquier caso, HL contiene la dirección de la matriz.

TXT FIJA TABLA DE MATRICES: BBAB, 12FE

Define una tabla de matrices de caracteres, la cual debe estar previamente almacenada en memoria.

Requisitos de entrada: DE contiene el primer carácter de la tabla, y HL la dirección de comienzo de la nueva tabla.

Condiciones de salida: si antes no había otra tabla de usuario definida, el acarreo estará a cero. Si antes había alguna tabla de modelos definida por el usuario, el acarreo estará a uno y, en este caso, A contendrá el primer carácter de la tabla antigua, y HL la dirección de la tabla antigua.

TXT IMPONE MATRIZ: BBA8, 12F2

Define una matriz (modelo) de carácter.

A la entrada, A contiene el código que se quiere asignar al carácter, y HL la matriz que se utilizará para definir el modelo.

A la salida, el acarreo será uno si el carácter es definible por el usuario, y será cero si no es definible.

TXT ESCR CARACTER: BDD3, 134B

Es una indirección. Escribe un carácter en la pantalla.

Requisitos de entrada: A contiene el carácter a escribir; H la columna física, y L la fila física donde se escribirá el carácter.

TXT ESCRIBE CARACTER: BB5D, 1335

Escribe un carácter en la posición actual del cursor y en la ventana actual. Los códigos de control serán ignorados y sólo se escribirá su carácter asociado.

A la entrada, A contiene el código del carácter.

TXT SALIDA: BB5A, 13FE

Envía un carácter o un código de control a la unidad de video (VDU) de texto.

A la entrada, A contiene el código a enviar.

TXT CODIGO CARACTER: BDD9, 140A

Es una indirección. Manda un carácter o un código de control. Los códigos de control serán obedecidos.

A la entrada, A contiene el código del carácter.

TXT LEE CAR: BDD6, 13BE

Lee un carácter de la pantalla. Es una indirección.

Requisitos de entrada: H contiene la columna física desde donde leer, y L la fila física.

Condiciones de salida: si se encontró algún carácter identificable, el acarreo estará a uno y A contendrá el código del carácter leído. Si no se pudo reconocer ningún carácter, el acarreo aparecerá a cero.

TXT ADMISION DE GRAFICOS: BB63, 13A8

Habilita o deshabilita la opción de caracteres gráficos.

A la entrada, $A=0$ para activar la opción gráficos y $A \neq 0$ para desactivarla.

CPC6128		CPC464
B6B5	Cauce actual	B20C
B6B6-B6C3	Datos del cauce 0	B20D-B21B
B6C4-B6D1	Datos del cauce 1	B21C-B22A
B6D2-B6DF	Datos del cauce 2	B22B-B239
B6E0-B6ED	Datos del cauce 3	B23A-B248
B6EE-B6FB	Datos del cauce 4	B249-B257
B6FC-B709	Datos del cauce 5	B258-B266
B70A-B717	Datos del cauce 6	B267-B275
B718-B725	Datos del cauce 7	B276-B284
B726	Fila actual	B285
B727	Columna actual	B286
B728	Tipo de desplazamiento actual	B287
B729	Borde superior actual	B288
B72A	Borde izquierdo actual	B289
B72B	Borde inferior actual	B28A
B72C	Borde derecho actual	B28B
B72D	Cuenta de desplazamiento actual	B28C
B72E	Flag del cursor actual	B28D
B72E	*Flag* de activación del cursor y de la pantalla actual	B28E
B72F	Pluma actual	B28F
B730	Papel actual	B290
B731-B732	Conexión para modo de impresión	B291-B292
B733	*Flag* de escritura de gráficos	B293
B734	Código de la primera matriz de la RAM	B294
B735	*Flag* de la matriz	B295
B736-B737	Dirección de la matriz de la RAM	B296-B297
B738-B757	Modelo elegido	B298-B2B7
B758	Contador de parámetros	B2B8
B759	Código de control	B2B9
B75A-B7C2	Parámetros de control	B2BA-B2C2
B763-B7C2	Tabla de saltos de control	B2C3-B322

La unidad de video para gráficos

GRA INICIALIZA: BBBA, 15A8

Inicializa la unidad de video para gráficos, indirecciones y variables. No hay requisitos.

GRA REINICIALIZA: BBBD, 15D7

Reinicializa las indirecciones de la VDU de gráficos. No hay requisitos.

GRA FIJA ORIGEN: BBC9, 160E

Determina el origen de coordenadas del cursor de gráficos.
Requisitos de entrada: DE contiene la coordenada X, y HL la coordenada Y.

GRA ANCHURA VENTANA: BBCF, 16A5

Define los márgenes derecho e izquierdo de la ventana de gráficos.
A la entrada, DE y HL contienen las coordenadas X estándar de los márgenes. El menor de ellos se tomará como margen izquierdo.

GRA ALTURA VENTANA: BBD2, 16EA

Define los bordes superior e inferior de la ventana de gráficos.
A la entrada, DE y HL contienen las coordenadas Y estándar de los bordes. El menor de ellos se tomará como borde inferior.

GRA LIMPIA VENTANA: BBDB, 1736

Borra la ventana de gráficos rellenándola con la tinta del papel asignado a gráficos.
No hay requisitos de entrada.

GRA COLOR PLUMA: BBDE, 1767

Define una tinta para la pluma de gráficos.
A la entrada, A contiene el número de tinta.

GRA COLOR PAPEL: BBE4, 176E

Define una tinta para el papel de gráficos.
A la entrada, A contiene el número de tinta.

GRA POSICION CURSOR: BBC6, 1606

Examina la posición del cursor de gráficos.
A la salida, DE contiene la coordenada X absoluta, y HL la coordenada Y.

GRA OBTIENE ORIGEN: BBCC, 161C

Examina la posición del origen de coordenadas.
A la salida, DE contiene la coordenada X estándar del origen, y HL la coordenada Y.

GRA OBTIENE ANCHURA: BBD5, 1717

Obtiene los márgenes derecho e izquierdo de la ventana de gráficos.
A la salida, DE contiene la coordenada X estándar del borde izquierdo, y HL la coordenada X del borde derecho.

GRA OBTIENE ALTURA: BBD8, 172D

Obtiene los límites superior e inferior de la ventana de gráficos.
A la salida, DE contiene la coordenada Y estándar del límite superior, y HL la coordenada Y del límite inferior.

GRA EXAMINA PLUMA: BBE1, 1775

Obtiene el color actual de la pluma de gráficos.
A la salida, A contiene el número de tinta.

GRA EXAMINA PAPEL: BBE7, 177A

Obtiene el color del papel de gráficos.
A la salida, A contiene el número de tinta.

GRA MOV ABSOLUTO: BBC0, 15FE
GRA MOV RELATIVO: BBC3, 15FB

Mueven el cursor a una posición en coordenadas absolutas y relativas, respectivamente.

A la entrada, DE contiene la coordenada X, y HL la coordenada Y.

GRA PUNTO ABSOLUTO: BBEA, 1783
GRA PUNTO RELATIVO: BBED, 1780

Activan un *pixel* en una posición en coordenadas absolutas y relativas, respectivamente.

A la entrada, DE contiene la coordenada X, y HL la coordenada Y.

GRA PUNTO: BDDC, 1786

Es una indirección. Dibuja un punto.

A la entrada, DE contiene la coordenada X, y HL la coordenada Y.

GRA TEST PUNTO ABS: BBF0, 1797
GRA TEST PUNTO REL: BBF3, 1794

Obtienen la tinta de un *pixel* en coordenadas absolutas y relativas, respectivamente.

A la entrada, DE contiene la coordenada X del punto, y HL la coordenada Y.

A la salida, A contiene el número de tinta.

GRA TEST PUNTO: BDDF, 179A

Es una indirección, igual que TEST PUNTO ABS, pero, a la salida, A contiene la tinta decodificada.

GRA LINEA ABSOLUTA: BBF6, 17A9
GRA LINEA RELATIVA: BBF9, 17A6

Dibujan una línea desde la posición actual del cursor de gráficos hasta la posición indicada, en coordenadas absolutas o relativas.

A la entrada, DE contiene la coordenada X del punto final, y HL la coordenada Y.

GRA LINEA: BBE2, 17B4

Es una indirección. Misma función y requisitos que GRA LINEA ABSOLUTA.

GRA ESCRIBE CARACTER: BBFC, 1940

Escribe un carácter en la posición actual del cursor de gráficos.
A la entrada, A contiene el carácter a escribir.

CPC6128	GRAFICOS	CPC464
B693-B694	Coordenada X del origen	B328-B329
B695-B696	Coordenada Y del origen	B32A-B32B
B697-B698	Posición X	B32C-B32D
B699-9A	Posición Y	B32E-B32F
B68B-9C	Izquierda de la ventana	B330-B331
B69D-9E	Derecha de la ventana	B332-B333
B69F-B6A0	Borde superior de la ventana	B334-B335
B6A1-A2	Borde inferior de la ventana	B336-B337
B6A3	Pluma codificada	B338
B6A4	Papel codificado	B339
B6A5-AC	Copia de matriz	B33A-B341
B6AD-AE	Coordenada X elegida	B342-B343
B6AF-B6B0	Coordenada Y elegida	B344-B345
B6B2		no existe
B6B3		no existe
B6B4	Modo de gráficos actual	no existe

El monitor de teclado

TCL INICIALIZACION: BB00, 1B5C

Inicializa completamente el monitor de teclado, variables, indirecciones y *buffers*.
No hay requisitos de entrada.

TCL REINICIALIZACION: BB03, 1B98

Reinicializa las indirecciones y los *buffers*. No hay requisitos.

TCL LEE TECLA: BB1B, 1CE1

Toma un código del *buffer* de teclado si hay alguno almacenado.
Condiciones de salida: Si hay alguno almacenado, el acarreo estará a uno, y A contendrá el carácter o la palabra clave de expansión. Si no hay ninguno, el acarreo estará a cero.

TCL ESPERA TECLA: BB18, 1CDB

Espera hasta que haya algún código en el *buffer* del teclado.
A la salida, el acarreo estará a uno, y A contendrá el carácter o palabra clave de expansión.

TCL LEE CARACTER: BB09, 1BC5

Intenta tomar un carácter del *buffer* del teclado; si no hay ninguno disponible, no espera.
A la salida, si hay algún código de carácter disponible, el acarreo estará a uno, y A contendrá el carácter. Si no hay ningún código, acarreo a cero.

TCL ESPERA CARACTER: BB06, 1BBF

Espera hasta que haya algún carácter disponible en el *buffer* del teclado.
A la salida, A contiene el carácter, y el acarreo estará a uno.

TCL TECLADO LIBRE: BD3D, 1BFE

Espera hasta que no haya ninguna tecla pulsada y después regresa.
A la salida, A contiene el código de la última tecla pulsada.

TCL REGRESA CARACTER: BB0C, 1BFA

Devuelve un carácter al *buffer* del teclado, de forma que pueda leerse más tarde.
A la entrada, A contiene el carácter a devolver.

TCL ASIGNA BUFFER: BB15, 1C04

Se define y asigna un *buffer* para cadenas de expansión.

Requisitos de entrada: DE contiene la dirección del *buffer*, y HL la longitud del *buffer*.

Condiciones de salida: Si el *buffer* está bien, el acarreo estará a uno; si es demasiado corto, el acarreo estará a cero.

TCL PONE EXPANSION: BB0F, 1C46

Impone la cadena de expansión asociada con una palabra clave de expansión.

Requisitos de entrada: B contiene la palabra clave de la expansión; C la longitud de la cadena, y HL la dirección de la cadena.

A la salida, el acarreo quedará a uno si la expansión es correcta.

TCL TOMA EXPANSION: BB12, 1CB3

Toma un carácter de una cadena de expansión. La cadena está numerada desde cero.

A la entrada, A contiene una clave de expansión y L el número del carácter a leer.

A la salida, el acarreo estará a uno si se encontró el carácter y A contiene el carácter. En caso contrario, el acarreo estará a cero.

EXAMINA TECLA: BB1E, 1E45

Examina si está pulsada una tecla en particular.

A la entrada, A contiene el número de la techa.

A la salida, el *flag* Z estará a cero si la tecla está pulsada y a 1 si no lo está.

TCL TOMA ESTADO: BB21, 1D38

Comprueba los estados de FIJA MAY, CONTROL FIJA MAY.

Condiciones de salida: L contiene el estado de CONTROL+FIJA MAY; H el estado de FIJA MAY. Si están activados, el contenido será &FF; en caso contrario, el contenido será &00.

TCL PONE ESTADO: BD3A, 1D3C

Fija los estados de FIJA MAY y de CONTROL+FIJA MAY.

A la entrada, H contiene el estado de FIJA MAY, y L el estado de CONTROL+FIJA MAY. &FF significa activación de la función, mientras que &00 indica desactivación.

TCL ESTADO JOYSTICK: BB24, 1DE5

Comprueba el estado del *joystick*.

Condiciones de salida: A y H contienen el estado del *joystick* 0, y L el estado del *joystick* 1. El byte de estado se configura de esta forma.

Bit 0:	Arriba
Bit 1:	Abajo
Bit 2:	Izquierda
Bit 3:	Derecha
Bit 4:	Fuego 2
Bit 5:	Fuego 1
Bit 6:	Botón de repuesto
Bit 7:	Siempre a cero

Si un bit está a uno, indica que el botón está pulsado.

PON TRASLADO: BB27, 1ED8

Determina el código o clave generada al pulsar una tecla sin que estén pulsadas CONTROL ni MAY.

A la entrada, A contiene un número de tecla, y B el código o clave. Las palabras claves con un uso especial son:

&80-&9F:	Palabras claves de expansión
&E0-&FC:	Edición, movimiento del cursor y rupturas
&FD:	FIJA MAY. Funciona en modo *toggle* (alternado)
&FE:	CONTROL+FIJA MAY. Funciona en modo *toggle*
&FF:	Se ignora

TCL TRASLADO CON MAY: BB2D, 1EDD

Determina el código, o clave, generado al pulsar una tecla a la vez que se pulsa MAY.

A la entrada, A contiene un número de tecla, y B el nuevo significado.

TCL TRASLADO CON CONTROL: BB33, 1EE2

Determina el código generado al pulsar una tecla a la vez que CONTROL.

A la entrada, A contiene un número de recla, y B el significado.

TCL TOMA TRASLADO: BB2A, 1EC4

Obtiene la interpretación de una tecla cuando se pulsa sin MAY ni CONTROL.

A la entrada, A contiene el número de tecla.

A la salida, A contiene la interpretación o palabra clave.

TCL OBTIENE MAY: BB30, 1EC9

Obtiene la interpretación de una tecla cuando se pulsa a la vez que MAY.

A la entrada, A contiene el número de tecla.

A la salida, A contiene la interpretación.

TCL OBTIENE CONTROL: BB63, 1ECE

Obtiene la interpretación de una tecla cuando se pulsa a la vez que CONTROL.

A la entrada, A contiene el número de tecla.

A la salida, A contiene el significado.

TCL IMPONE REPETICION: BB39, 1E34

Determina si se asigna a una tecla la repetición.

A la entrada, A contiene el número de tecla; B contiene &FF, si se permite la repetición, o &00 si no se permite.

TCL CONOCE REPETICION: BB3C, 1E2F

Comprueba si se permite repetición a una tecla.

A la entrada, A contiene el número de la tecla.

A la salida, el *flag* Z estará a cero si se permite la repetición, o a 1 si no se permite.

TCL IMPONE RETARDO: BB3F, 1DF6

Asigna a todas las teclas el retardo de comienzo y el período de repetición.

A la entrada, H contiene el retardo del comienzo, y L el nuevo período de repetición, medido en 1/50 segundos.

TCL CONOCE RETARDO: BB42, 1DF2

Obtiene el retardo de comienzo y el período de repetición.

A la salida, H contiene el retardo, y L el período de repetición.

TCL INHIBE RUPTURAS: BB48, 1E08

Deshabilita el mecanismo de ruptura, es decir, la ejecución de los sucesos *break*.

No hay requisitos.

TCL PERMITE RUPTURAS: BB45, 1DFA

Habilita el mecanismo de ruptura.

A la entrada, DE contiene la dirección de la rutina asociada al suceso de ruptura, y C la dirección de la ROM para esa rutina.

TCL GENERA RUPTURA: BB4B, 1E19

Genera un suceso *break* si éstos están habilitados. No hay requisitos.

La palabra clave &EF se coloca en el *buffer* del teclado, de forma que, cuando éste se lea, provoque la generación del suceso.

TCL EXAMINA ESC: BDEE, 1DB8

Comprueba si la tecla ESC está pulsada.

Requisitos de entrada: Interrupciones inhibidas; C contiene el estado de las teclas MAY y CONTROL.

ESPACIO DE TRABAJO DEL MONITOR DE TECLADO

CPC6128		CPC464
B416-B465	Tabla 1 de código/tecla: tecla MAY sin pulsar	B34C-B39B
B4E6-B535	Tabla 2 de código/tecla: tecla MAY pulsada	B39C-B3EB
B536-B585	Tabla 3 de código/tecla: tecla CONTROL pulsada	B3EC-B34B
B586-B6D5	Tabla de control de repeticiones	B34C-B49B
B590-B627	*Buffer* del teclado (cuidado con solapamientos)	B446-B4DD
B628	Puntero del *buffer*	B4DE
B629	Señal de la cadena actual	B4DF
B62A	Devuelve carácter	B4E0
B62B/C	Comienzo del *buffer* de cadenas	B4E1/2
B62D/E	Final del *buffer* de cadenas	B4E3/4
B62F/30	Comienzo del espacio libre en el *buffer* de cadenas	B4E5/6
B631/2	Estado FIJA/MAY del teclado	B4E7/8
B631/2	Velocidad de repetición	B4E9
B634	Pausa o retardo de repetición	B4EA
B635-B63E	Mapa 3	B4EB-B4F4
B63F-B648	Mapa 1	B4F5-B4FE
B649-B652	Mapa 2	B4FF-B508
B653	Cuenta de repetición	B509
B654	Número del mapa de bytes	B50A
B655	Mapa de máscaras de bit	B50B
B656	*Flag* para permitir rupturas	B50C
B657-B65D	Bloque de ruptura de sucesos	B50D-B513
B65E-B685	*Buffer* del teclado	B514-B53B
B686	Espacio libre del *buffer*+1	B53C
B687	Puntero de entrada	B53D
B688	Entradas del *buffer*+1	B53E
B689	Puntero de salida	B53F
B68A	Entradas del *buffer*	B540
B68B/C	Puntero de la tabla 1 de tecla/código	B541/2
B68D/E	Puntero de la tabla 2 de tecla/código	B543/4
B68F/90	Puntero de la tabla 3 de tecla/código	B545/6
B691/2	Puntero de la tabla de control de repeticiones	B547/8

El monitor de disco y de cassette

Modo cassete (|TAPE, |TAPE.IN, |TAPE. OUT)

CAS INICIALIZA: BC65, 24BC

Inicializa el monitor de cassette, los cauces *(streams)*, la velocidad de escritura y activa los mensajes de ayuda. No hay requisitos.

CAS VELOCIDAD: BC68, 24CE

Define la velocidad de escritura y la precompensación a aplicar.

Requisitos de entrada: HL contiene la duración en microsegundos de medio período de un bit a nivel bajo; es válido cualquier valor entre 130 y 480. A debe contener la precompensación.

CAS MENSAJES: BC6B, 24E1

Activa o desactiva los mensajes de ayuda.

A la entrada, A=0 para activar mensajes o $A \neq 0$ para desactivarlos.

CAS ARRANCA MOTOR: BC6E, 2BBB

Enciende el motor del cassette y espera dos segundos hasta que se estabilice la velocidad.

A la salida el acarreo estará a uno si el motor empezó a funcionar; estará a cero en caso de que se pulsara ESC. En cualquier caso, A contiene el estado interior del motor.

CAS PARA MOTOR: BC71, 2BBF

Desconecta el motor del cassette.

A la salida, A contiene el estado anterior del motor. El acarreo estará a uno si se realizó la desconexión y a cero si se pulsó ESC.

CAS RESTAURA MOTOR: BC74, 2BC1

Restaura el estado anterior del motor.
A la entrada, A debe contener el estado previo.
A la salida, el acarreo estará a uno a menos que se pulse ESC.

DCAS ABRE ENTRADA: BC77, 24E5

Abre un fichero de entrada.

Requisitos de entrada: B contiene la longitud del nombre del fichero; HL contiene la dirección del nombre, y DE contiene la dirección de un área de 2K que será el *buffer* para la carga.

Condiciones de salida: HL contiene la dirección del *buffer* donde está la cabecera del fichero; DE contiene la dirección de los datos según la cabecera; BC contiene la longitud del fichero según la cabecera, y A contiene el tipo de fichero.

Estas condiciones de salida sólo son ciertas si el acarreo está a uno y el *flag* Z a cero. Si se pulsó ESCape, el acarreo y Z quedarán a uno. En el resto de los casos, el acarreo y Z quedarán a cero.

DCAS CIERRA ENTRADA: BC7A, 2550

Cierra un fichero de entrada. A la salida, el acarreo estará a uno si se realizó la acción.

DCAS ABANDONA ENTRADA: BC7D, 257F

Abandona un fichero de entrada. No hay requisitos.

DCAS LEE CARACTER: BC80, 25A0

Lee un carácter del fichero de entrada.

A la salida, A contiene el código del carácter, lo cual queda indicado si el acarreo (C) está a uno y el *flag* de cero (Z) está a cero. Si se encontró el final del fichero: Z=0 y C=0. Si se pulsó ESC: C=0 y Z=1.

DCAS ENTRADA DIRECTA: BC83, 2618

Lee un fichero completo.
A la entrada, HL contiene la dirección donde se almacenará el fichero.

Condiciones de salida: el fichero se leyó correctamente, C=1, Z=0 y HL contiene la dirección de entrada según la cabecera. Si el fichero no estaba abierto, C=0 y Z=0. Si se pulsó ESC, C=0 y Z=1.

DCAS RETORNO: BC86, 2607

Devuelve al fichero de entrada el último carácter leído. No hay requisitos.

DCAS EXAMINA FINAL: BC89, 2603

Comprueba si se ha alcanzado el final del fichero.

A la salida, C=1 y Z=0 si aún no se ha llegado al final del fichero; C=0 y Z=0 si se ha alcanzado; C=0 y Z=1 si se pulsó ESC.

DCAS ABRE SALIDA: BC8C, 24FE

Abre un fichero de salida.

Requisitos de entrada: B contiene la longitud del nombre del fichero; HL la dirección del nombre y DE la dirección de un área de 2K que actúa como *buffer*.

Condiciones de salida: Si el fichero ya estaba abierto, C=0; si el fichero se abre correctamente, C=1, y HL contiene la dirección donde está la cabecera.

DCAS CIERRA SALIDA: BC8F, 256D

Cierra un fichero de salida y lo graba.

A la salida, C=1 y Z=0 si se realizó correctamente la operación; C=0 y Z=0 si el fichero no estaba abierto; C=0 y Z=1 si se pulsó ESC.

DCAS ABANDONA SALIDA: BC92, 2599

Abandona un fichero de salida. No hay requisitos.

DCAS SALIDA CARACTER: BC95, 25C6

Escribe un carácter en el fichero de salida.
A la entrada, A contiene el carácter a escribir.
A la salida, C=1 y Z=0 si se realizó bien la operación; C=0 y Z=0 si el fichero no estaba abierto; C=0 y Z=1 si se pulsó ESC.

DCAS SALIDA DIRECTA: BC98, 2653

Escribe un fichero completo.
Requisitos de entrada: HL contiene la dirección donde están los datos a escribir; DE la longitud de los datos; BC la dirección de entrada (para el encabezamiento) y A el tipo de fichero (para el encabezamiento).
Condiciones de salida: Si el fichero se escribió, C=1 y Z=0; si el fichero no estaba abierto, C=0 y Z=0; si se pulsó ESC, entonces C=0 y Z=1.

DCAS CATALOGA: BC9B, 2692

Lista el contenido de una cinta. A la entrada, DE contiene la dirección de un *buffer* de 2 Kbytes para la información.
A la salida, el acarreo estará a uno si se realizó bien la operación. Además, C=0 y Z=0 si había algún fichero abierto, y C=0, Z=1 si se produjo error.

CAS ESCRIBE: BC9E, 29AF

Escribe una cadena de caracteres en la cinta.
A la entrada, HL contiene la dirección de los datos a escribir: DE la longitud de los datos, y A el carácter de sincronismo a escribir al final de la antegrabación.
A la salida, el acarreo estará a uno si todo fue bien, pero si el acarreo está a cero, entonces ocurrió algún error y A contiene el código del error.
Los códigos de error de escritura son:

0: Se pulsó ESCape.
1: Error de escritura.

CAS LEE: BCA1, 2946

Lee un registro o cadena de la cinta.

A la entrada, HL contiene la dirección donde se guardarán los datos; DE la longitud de los datos a leer, y A el carácter de sincronismo esperado al final de la antegrabación.

A la salida, acarreo a uno si todo fue bien, pero acarreo a cero si hubo error, y entonces A contiene un código de error.

Los códigos de error de lectura son:

0: Se pulsó ESCape.
1: Bit demasiado largo.
2: Comprobación fallida de CCS.

CAS COMPARA: BCA4, 29C1

Compara una grabación con el contenido de la memoria.

Requisitos de entrada: HL contiene la dirección de los datos a comparar; DE la longitud de los datos, y A el carácter de sincronismo esperado.

Condiciones de salida: Si la comparación es correcta, acarreo (C) a uno; si ocurrió algún error, C=0, y A contiene un código de error.

Los códigos de error de comparación son:

0: Se pulsó ESCape.
1: Bit demasiado largo.
2: Comprobación fallida de CCS.
3: Los datos de la cinta son diferentes a los de la memoria.

ESPACIO DE TRABAJO PARA EL CASSETTE

CPC6128		CPC464
B118	*Flag* para CAS mensajes (0 habilita mensajes)	B800
B119	Contador de columnas de pantalla (para los mensajes)	B801
	Bloque de ficheros de entrada	
B11A	Estado del fichero	B802
B11B/C	Dirección del *buffer*	B803/4
B11D/E	Puntero del *buffer*	B805/6
B11F/2E	Nombre del fichero	B807/16
B12F	Número de bloques	B817
B130	*Flag* para EOF (final del fichero)	B818

CPC6128		CPC464
B131	Tipo de fichero	B819
B132/3	Bytes en el *buffer*	B81A/B
B134/5	Dirección de escritura de datos	B81C/D
B136	*Flag* de primer bloque	B81E
B137/8	Longitud de datos	B81F/20
B139/A	Dirección de ejecución	B821/2
B13B/5D	Sin definir	B823/46

Bloque de ficheros de salida

B15F	Estado del fichero	B847
B160/1	Dirección del *buffer*	B848/9
B162/3	Puntero del *buffer*	B84A/B
B164/73	Nombre del fichero	B84C/5B
B174	Número de bloques	B85C
B175	*Flag* para EOF (final del fichero)	B85D
B176	Tipo de fichero	B85E
B177/8	Bytes en el *buffer*	B85F/60
B179/A	Dirección de lectura de datos	B861/2
B17B	*Flag* de primer bloque	B863
B17C/D	Longitud de datos	B864/5
B17E/F	Dirección de comienzo de ejecución	B866/7
B180/A3	Sin definir	B868/8B

Copia del encabezamiento

B1A4/B3	Nombre del fichero	B88C/9B
B1B4	Número de bloque	B89C
B1B5	*Flag* de último bloque	B89D
B1B6	Tipo de fichero	B89E
B1B7/8	Longitud de datos	B89F/0
B1B9/A	Dirección de los datos	B8A1/2
B1BB	*Flag* de primer bloque	B8A3
B1BC/D	Longitud total de datos	B8A4/5
B1BE/F	Dirección de comienzo de ejecución	B8A6/7
B1C0/E3	Sin definir	B8A8/CB

General

B1E4	*Flag* para mensajes de ayuda	B8CC
B1E5	Carácter de sincronismo	B8CD
B1E6/7	Temporización	B8CD/F
B1E8	Temporización	B8D0
B1E9	Precompensación	B8D1
B1EA	Velocidad	B8D2
B1EB/C	Temporización	B8D3/4

Modo disco (|DISC, |DISC.IN, |DISC.OUT)

En este modo sólo activarán la unidad de disco las siguientes rutinas:

BC77	DCAS	ABRE ENTRADA
BC7A	DCAS	CIERRA ENTRADA
BC7D	DCAS	ABANDONA ENTRADA
BC80	DCAS	LEE CARACTER
BC83	DCAS	ENTRADA DIRECTA
BC86	DCAS	RETORNO
BC89	DCAS	EXAMINA FINAL
BC8C	DCAS	ABRE SALIDA
BC8F	DCAS	CIERRA SALIDA
BC92	DCAS	ABANDONA SALIDA
BC95	DCAS	SALIDA CARACTER
BC98	DCAS	SALIDA DIRECTA
BC9B	DCAS	CATALOGA

Todas ellas tienen los mismos requisitos de entrada y las mismas condiciones de salida expuestas anteriormente para el modo cassette.

El encabezamiento de los ficheros se guarda a partir de la dirección A752, siendo las direcciones con los contenidos más importantes las expuestas al final de la descripción del monitor de la ROM de disco.

Es importante destacar la necesidad de atenerse al formato de los nombres de ficheros impuesto para disco. Según esto, al abrir un fichero desde código máquina, ya sea de salida o de entrada, el nombre debe tener la forma XXXXXXXX.XXX, rellenando con espacios (CHR$(32)) el resto hasta tener dieciséis caracteres.

Sin embargo, en el *buffer* de encabezamiento, el punto que separa los tres últimos caracteres se sobreentiende y no aparecerá almacenado. El byte de "tipo de fichero" se interpreta así:

Bit 0	Si es 1, el fichero está protegido.
Bits 1-3	000: BASIC
	001: BINARIO
	010: Volcado de pantalla
	011: ASCII (CP/M)
Bits 4-7	000: No es ASCII
	001: ASCII
	Resto: No definido

El monitor de sonido

REINICIALIZA SONIDO: BCA7, 1FE9

Inicializa el monitor de sonido, borra todas las colas de sonido y vacía los registros del generador programable de sonido (PSG).

No hay requisitos.

COLA SONIDO: BCAA, 2114

Intenta añadir un sonido a la cola de sonidos de uno o varios canales. No lo conseguirá si la cola estaba llena.

A la entrada, HL contiene la dirección de un bloque que define un sonido.

A la salida, el acarreo a uno si el sonido se añadió a la cola y a cero si la cola estaba llena. Véase capítulo 11 para la interpretación del bloque de definición.

EXAMINA COLA: BCAD, 21CE

Comprueba el estado de un canal, incluido el espacio libre en la cola.

A la entrada, A contiene el bit propio del canal a examinar.

Bit 0 a uno: canal A
Bit 1 a uno: canal B
Bit 2 a uno: canal C

A la salida, A contiene el estado del canal. Véase capítulo 11 para la interpretación del estado.

FORMA SUCESO SONIDO: BCB0, 21EB

Actualiza un suceso que se ejecutará cuando la cola de sonido se vacíe.

Requisitos de entrada: A contiene el bit propio del canal y HL contiene la dirección de la rutina del suceso asociado.

PARA SONIDO: BCB6, 2050

Detiene inmediatamente la ejecución de sonidos.

A la salida, el acarreo estará a uno si había algún sonido ejecutándose; acarreo a cero, si no había ningún sonido.

CONTINUA SONIDO: BCB9, 206B

Permite continuar la ejecución de los sonidos que la rutina anterior había retenido. No hay requisitos.

EJECUTA SONIDO: BCB3, 21AC

Permite la producción de sonido únicamente a los canales que se especifiquen.

A la entrada, A contiene los bits propios de los canales que se desea.

ENVOLVENTE VOLUMEN: BCBC, 249E

Define una de las 15 envolventes de volumen programables.

A la entrada, A contiene el número de envolvente que se va a definir; HL contiene la dirección de un bloque que contiene los datos de la envolvente.

A la salida, el acarreo estará a uno si se definió correctamente, y HL contiene la dirección del bloque más 16. Si el número de la envolvente es inválido, acarreo a cero. Véase capítulo 11 para la interpretación del bloque de datos de la envolvente.

ENVOLVENTE TONO: BCBF, 249A

Define una de las 15 envolventes de tono programables.

A la entrada, A contiene el número de envolvente, y HL la dirección del bloque que contiene los datos de la envolvente.

Las condiciones de salida son las mismas que ENVOLVENTE VOLUMEN.

DIRECCION ENVOLVENTE: BCC5, 24A6

Obtiene la dirección de un bloque definitorio de envolvente de volumen.

A la entrada, A contiene el número de la envolvente.

A la salida, el acarreo estará a uno si se encontró el bloque. Entonces, HL contendrá la dirección del bloque, y BC, la longitud de la envolvente. Si el número de la envolvente no es correcto, acarreo a cero.

DIRECCION ENVOLVENTE T: BCC5, 24AB

Obtiene la dirección de un bloque de datos de una envolvente de tono.

Los requisitos de entrada y condiciones de salida son los mismos que para DIRECCION ENVOLVENTE A.

ESPACIO DE TRABAJO DEL MONITOR DE SONIDO

CPC6128		CPC464
B1EC	*Flags* para nuevas entradas	B550
B1ED	*Flags* para la retención de canales activos	B551
B1EE	*Flags* para los canales activos	B552
B1EF	Contador del control de interrupciones	B553
B1F0	Contador de acciones pendientes	B554
B1F1-B1F7	Bloque de sucesos	B555-B55B
B1F8-B236	*Buffer* del canal A	B55C-B59A
B237-B275	*Buffer* del canal B	B59B-B5D9
B276-B2B4	*Buffer* del canal C	B5DA-B618
B2B5	Byte de activación del PSG	B619
B2B6-B3A5	Envolventes de volumen	B61A-B709
B3A6-B495	Envolventes de tono	B70A-B7F9

FORMATO DEL *BUFFER* DE UN CANAL RELATIVO A SU DIRECCION

&00	Número de canal
&01	Bit del canal
&02	Bit de acordes del canal
&03	*Flags* de estado del canal
&04	Llamadas para el establecimiento del tono si bit 0=1
&05	Contador 1 de interrupciones
&06	Contador 2 de interrupciones
&07	Llamadas para el establecimiento del volumen si bit 0=1

&08-&09	Duración negada (complementada)
&0A-&0B	Dirección de la envolvente de volumen
&0C	Número de secciones de la envolvente de volumen
&0D-&0E	Dirección de la sección de la envolvente de volumen

TABLA DE CARACTER DE CONTROL

0	14EB	Regreso inmediato. Sin acción	14E2
1	1335	TXT ESCRIBE CARACTER	1334
2	139B	TXT INHIBE CURSOR USUARIO	139A
3	1286	TXT ACTIVA CURSOR USUARIO	1289
4	0AE9	PNT FIJA MODO	0ACA
5	1940	GRA ESCRIBE CARACTER	1945
6	1459	TXT ACTIVA VDU	1451
7	14E1	COLA SONIDO con HL = 14D8 (pitido)	14D8
8	1519	Cursor izquierda	150A
9	151E	Cursor derecha	150F
10	1523	Cursor abajo	1514
11	1528	Cursor arriba	1519
12	154F	TXT LIMPIA VENTANA	1540
13	153F	Columna = borde izquierdo ventana	1530
14	12AB	TXT COLOR PAPEL	12AE
15	12A6	TXT COLOR PLUMA	12A9
16	155E	Borrar	154F
17	1599	Limpia ventana a la izquierda del cursor	158E
18	158F	Limpia ventana a la derecha del cursor	1584
19	1578	Limpia ventana encima del cursor	156D
20	1565	Limpia ventana debajo del cursor	1556
21	1452	TXT DESACTIVA VDU	144B
22	14EC	TXT FIJA MODO	14E3
23	0C55	PNT MODO GRAFICOS	0C49
24	12C6	TXT INVIERTE COLORES	12C9
25	150D	TXT IMPONE MATRIZ	1504
26	1501	TXT ACTIVA VENTANA	14F8
27	14EB	Regreso inmediato sin acción	14E2
28	14F1	PNT IMPONE TINTA	14E8
29	14FA	PNT IMPONE BORDE	14F1
30	1539	Cursor a esquina superior izquierda	152A
31	1543	POSICIONA CURSOR	1538

Indice alfabético

Acarreo, 46-47.
Acceso a la ROM de disco, 192-194.
Acción repetitiva, 120.
ACTIVA ROM INFERIOR, 25, 207.
ACTIVA ROM SUPERIOR 25, 193, 206.
AMSDOS, 17.
Area de datos del núcleo, 57.
Area de extensiones RST, 28, 208.
Area RST, 21-22.

BASIC, 17, 33, 36, 122, 128, 153, 159, 161-162, 171, 191.
Bloques de saltos para núcleos, 206.
Bloques de control de ficheros de salida, 137.
BORRA SINCRONO BARRIDO, 213.
Buffer, 112, 116, 128, 132, 137.
BUSY, 40.

CALL, 21, 23, 168, 191.
CAS ABANDONA ENTRADA, 132.
CAS ABANDONA SALIDA 134.
CAS ABRE ENTRADA, 131, 133.
CAS ABRE SALIDA, 131, 133.
CAS ARRANCA MOTOR, 130, 240.
CAS CATALOGA, 135.
CAS CIERRA ENTRADA, 132.
CAS CIERRA SALIDA, 134.
CAS COMPARA, 128, 136, 244.
CAS ENTRADA DIRECTA, 128, 133.
CAS ESCRIBE, 129, 135.
CAS EXAMINA FINAL, 133.
CAS INICIALIZA 38, 130, 240.
CAS LEE, 129, 136, 244.
CAS LEE CARACTER, 128, 132-133.
CAS MENSAJES, 129-130, 240.
CAS PARA MOTOR, 131, 240.
CAS RESTAURA MOTOR, 130, 241.
CAS RETORNO, 129, 133.
CAS SALIDA CARACTER, 134.
CAS SALIDA DIRECTA, 128, 135.
CAS VELOCIDAD, 130, 240.
Cauces, 92.
Cauces y ventanas, 61.

CINT, 161.
Código ASCII, 99, 171.
COLA SONIDO, 141-144, 247.
Color, 90.
Coma flotante, 159.
Compresión de pantalla, 62.
CONTINUA SONIDO, 144-145, 248.
Control de cursor, 86.
Control de pantalla, 67, 86.
Controlador CRT, 36.
Controlador de interrupciones, 46.
Copia del encabezamiento de ficheros de entrada, 200.
Copia del encabezamiento de ficheros de salida, 199.
CRT, 14-15, 60.

DATA, 17.
DCAS ABANDONA ENTRADA, 241.
DCAS ABANDONA SALIDA, 242.
DCAS ABRE ENTRADA, 241.
DCAS ABRE SALIDA, 242.
DCAS CATALOGA, 243.
DCAS CIERRA ENTRADA, 241.
DCAS CIERRA SALIDA, 242.
DCAS ENTRADA DIRECTA, 241.
DCAS ESCRIBE, 243.
DCAS EXAMINA FINAL, 242.
DCAS LEE CARACTER, 241.
DCAS RETORNO, 242.
DCAS SALIDA CARACTER, 243.
DCAS SALIDA DIRECTA, 243.
DE, 37-38, 41.
DEG, 163.
DEHL, 56.
DEHLC, 160.
DFXX, 152.
DIRECCION ENVOLVENTE, 249.
DIRECCION ENVOLVENTE T, 147, 249.
DIRECCION ENVOLVENTE V, 147.
DISC MENSAJES, 194.

EJECUTA SONIDO, 145, 248.
Entradas del bloque de saltos, 17.
ENVOLVENTE TONO, 146, 248.
ENVOLVENTE VOLUMEN, 145, 248.
ESCAPE, 122, 129.
ESCRIBE PIXEL, 67.
ESCRIBE SECTOR, 196.
Espacio de trabajo del monitor de sonido, 148.
Espacio de trabajo del monitor de teclado, 124.
Espacio de trabajo para el cassette, 137.
ESTADO ENTRADA D0, 198.
ESTADO ENTRADA D1, 199.
ESTADO SALIDA, 199.
ESTADO SALIDA D0, 199.
Estados del sistema, 16.
Etiquetas, 183.
EXAMINA COLA, 142, 144, 247.
EXAMINA TECLA, 235.
E/S ENTRADA D0, 198.
E/S ENTRADA D1, 199.
E/S SALIDA D0, 199.
E/S SALIDA D1, 199.

FIJA REINTENTOS, 197.
Firmware, 191.
FIX, 162.
Flag, 97, 132.
Flag de matriz, 93.
FOR, 170.
FORMA SUCESO SONIDO, 145, 247.
FORMATEA PISTA, 196.
Funciones de ruptura, 121.

Generador de sonido, 36.
GRA ALTURA VENTANA, 105, 230.
GRA ANCHURA VENTANA, 104, 230.
GRA COLOR PAPEL, 105, 230.
GRA COLOR PLUMA, 105, 230.
GRA ESCRIBE CARACTER, 97, 109, 233.
GRA EXAMINA PAPEL, 106, 231.
GRA EXAMINA PLUMA, 106, 231.
GRA FIJA ORIGEN, 104, 230.
GRA INICIALIZA, 38, 68, 104, 229.
GRA LIMPIA VENTANA, 105, 230.
GRA LINEA, 108, 233.
GRA LINEA ABSOLUTA, 108, 232.
GRA LINEA RELATIVA, 108, 232.

GRA MOV ABSOLUTO, 107, 232.
GRA MOV RELATIVO, 107, 232.
GRA OBTIENE ALTURA, 231.
GRA OBTIENE ANCHURA, 106, 231.
GRA OBTIENE ORIGEN, 106, 231.
GRA POSICION CURSOR, 105, 231.
GRA PUNTO, 107, 232.
GRA PUNTO ABSOLUTO, 107.
GRA PUNTO RELATIVO, 107.
GRA REINICIALIZA, 104, 230.
GRA TEST PUNTO, 108, 232.
GRA TEST PUNTO ABS, 232.
GRA TEST PUNTO REL, 108, 232.
GRA TEXT PUNTO, 108.
Gráficos, 63.
GUARDA REGISTRO, 197.

Hardware, 21, 35, 43, 151.
HIMEM, 17.
HL, 37-38, 50-53.

INICIA SIO, 198.
INICIALIZA DISCO, 195.
INF SALTO, 29, 33, 161.
INF SALTO HL, 28.
INHIBE ROM INFERIOR, 25, 207.
INHIBE ROM SUPERIOR, 25, 207.
INICIA BUFFERS ORDENES, 198.
INT, 162.
Intérprete de BASIC, 167.
Interrupciones, 38, 45, 52.
IN(C),C, 192.

JP(BC), 22.

LAM RAM, 32-33, 132.
LD A,(HL), 24.
LDDR, 27, 208.
LDIR, 27, 68, 77, 208.
LEE SECTOR, 195.
LIMPIA MODO, 67.
Load failed, 37-38.
LOCALIZA PISTA, 196.
LOCK, 117.
LOG, 160.

LLAMADA, 24, 30, 151, 193.
LLAMADA FIRMWARE, 197.
LLAMADA HL, 24, 30.
LLAMADA LATERAL, 23-24, 31, 152.
LLAMADA POR 0D3C Y 0D5B, 75.
LLAMADA POR 0D4F, 0D5B Y 0D6D, 75.
Llamadas diversas, 135.

Manejo de llamadas para matemáticas, 164.
Mapa, 111.
Mapa de E/S, 15.
Mapa de memoria, 13, 189.
Mapas de bits, 123.
Matriz de puertas de video, 46.
MAXAM, 17.
MC ASIGNA TINTAS, 41, 211.
MC BORRA TINTAS, 42, 210.
MC CARGA Y EJECUTA, 36, 209.
MC EJECUTA PROGRAMA, 36, 39, 46, 157, 209.
MC ENVIA CARACTER, 40, 210.
MC ESPERA BARRIDO, 42, 211.
MC ESPERA IMPRESORA, 39-40, 210.
MC IMPRESORA OCUPADA, 40, 210.
MC IMPRIME CARACTER, 40, 210.
MC INICIA IMPRESORA, 209.
MC INICIALIZA IMPRESORA, 38-39.
MC MODO PANTALLA, 42, 68, 211.
MC OFFSET PANTALLA, 42, 69, 211.
MC REGISTRO SONIDO, 43, 142, 211.
Mensajes, 129.
Modo disco, 246.
Monitor de cassette, 127, 240.
Monitor de la unidad de disco, 194.
Monitor de sonido, 141, 247.
Monitor de teclado, 111, 233.

NC AÑADE SINCRONO BARRIDO, 51, 213.
NC AÑADE SINCRONO LENTO, 52, 214.
NC AÑADE SINCRONO RAPIDO, 51, 213.

NC BORRA SINCRONO BARRIDO, 51.
NC BORRA SINCRONO LENTO, 52, 214.
NC BORRA SINCRONO RAPIDO, 51, 213.
NC BORRA SUCESO, 53, 212.
NC BUSCA ORDEN, 156.
NC DESINHIBE SUCESO, 55, 215.
NC EJECUTA PROGRAMA, 37.
NC EJECUTA SUCESO, 54, 214.
NC IGNORA SUCESO, 52-53, 214.
NC INHIBE SUCESO, 55, 215.
NC INIC SECUNDARIA, 155.
NC INICIALIZA SUCESO, 49-50, 212.
NC INICIALIZACION, 38.
NC INTRODUCE RSX 154.
NC NUEVO SINCRONO BARRIDO, 50, 213.
NC NUEVO SINCRONO RAPIDO, 51, 213.
NC PONE TIEMPO, 56, 215.
NC PRIORIDAD, 54-55, 208, 214.
NC REINICIALIZA NUCLEO, 37-38.
NC REINICIALIZA SINCRONOS, 53, 212.
NC ROM SECUNDARIA, 152, 156.
NC SELECCIONA ROM, 157.
NC SIGUIENTE SUCESO, 54-55, 212-213.
NC SUCESO, 50, 212.
NC SUCESO EJECUTADO, 54-55, 214.
NC TIEMPO, 56, 215.
Núcleo, 45.

OBTIENE ESTADO, 197.
OUT(C),C, 192.

Palabras reservadas, 168-170.
PARAR SONIDO, 145, 248.
Parámetros de pantalla, 62.
PARPADEO DE COLORES, 72, 74.
PEEK, 14.
Periféricos externos, 16.
Pixel, 70-71.
PNT BASE Y OFFSET, 217.
PNT BORRA PANTALLA, 67-68, 216.
PNT BYTE ANTERIOR, 71, 218.
PNT CODIFICA TINTA, 72, 219.
PNT COMPRIME, 78, 221.
PNT COORDENADA PUNTOS, 70, 218.
PNT DECODIFICA TINTA, 73, 219.
PNT DESPLAZA AREA, 77, 88, 221.
PNT DESPLAZA PANTALLA, 77, 220.
PNT ESCRIBE PIXEL, 79, 81, 222.
PNT EXAMINA MODO, 69, 217.
PNT EXAMINA FLASH, 74, 220.
PNT EXPANDE, 78, 221.
PNT FIJA FLASH, 74, 220.
PNT FIJA MODO, 68, 217.
PNT HORIZONTAL, 80, 222.
PNT IMPONE BASE, 217.
PNT IMPONE BASE RAM, 70.
PNT IMPONE BORDE, 73, 219.
PNT IMPONE OFFSET, 68-69, 217.
PNT IMPONE TINTA, 73, 219.
PNT INICIALIZA, 38, 67, 216.
PNT INVIERTE CARACTER, 77, 88.
PNT INVIERTE COLORES, 220.
PNT LEE PIXEL, 67, 81, 222.
PNT LIMITES PANTALLA, 70, 218.
PNT LIMPIA MODO, 68.
PNT LINEA ANTERIOR, 71.
PNT LOCALIZA RAM PANTALLA, 69-70, 218.
PNT MODO GRAFICOS, 79, 221.
PNT OBTIENE BORDE, 74, 219.
PNT OBTIENE TINTA, 74, 219.
PNT PIXEL, 80, 222.
PNT POSICIONA COORDENADAS, 70, 96, 218.
PNT REINICIALIZA, 37, 216.
PNT RELLENA BYTE, 76-77, 220.
PNT RELLENA CAJA, 76-220.
PNT REINICIALIZA, 67.
PNT SIGUIENTE BYTE, 71, 76, 218.
PNT SIGUIENTE LINEA, 71, 76, 96, 219.
PNT VERTICAL, 81, 222.

PON TRASLADO, 236.
POP HL, 47.
PRINT, 167, 170.
Programas de soporte, 171.
PSG, 141-142.

RAD, 163.
RAM, 16, 21-24, 35, 37.
RAM de pantalla, 59.
RAM superior, 24.
REINICIALIZA SONIDO, 36, 38, 143-144, 247.
REINICIALIZACION PRINCIPAL, 35.
RESTAURA, 26.
RESTAURA BLOQUE SALTOS, 38.
RESTAURA ROM, 207.
ROM, 13-16, 21-26, 37.
ROM actual, 27, 207.
ROM anterior, 27, 155, 208.
ROM de disco, 191.
ROM inferior, 35.
ROMDIS, 151.
ROMEM, 151.
RST, 23-24, 36.
RST28, 33.
RST8, 33.
RSX, 193.
Rutinas de acceso a la impresora para el 6128, 209.
Rutinas de la RAM, 21.
Rutinas de teclado, 113.
Rutinas de texto, 99.
Rutinas del *firmware* en el 6128, 200.
Rutinas del núcleo, 56.
Rutinas específicas del 6128, 191.
Rutinas llamadas por PNT BORRA PANTALLA, 75.
Rutinas MC, 41.
Rutinas RAM del bloque de saltos, 25.

SALIDA, 55.
Salida de textos, 95.
Salto BC, 22.
Salto HL, 22, 24, 30, 154.
Salto HL lateral, 23, 31.
Salto INF, 193.
Salto ROM inferior, 32, 161.
SELECCIONA FORMATO, 195.
SELECCIONA ROM, 26, 193, 207.
SHIFT, 112.
SIGUIENTE SUCESO, 54.
SIO, 199.
Sistema de pantalla, 59.
Sistema de parpadeo, 74.
Sistema de sucesos, 48.
Sistema operativo, 9.
Software, 43, 151.
Soporte del BASIC, 159.
SPC, 170.
STEP, 170.
STROBE, 40.
Sucesos síncronos, 53.

Tabla de caracteres de control, 98.
Tabla de máscaras, 69.
Tablas de teclas y códigos, 118.
Tablas tecla/código, 123.
TCL ASIGNA BUFFER, 116, 235.
TCL CONOCE REPETICION, 120, 238.
TCL CONOCE RETARDO, 120, 238.
TCL ESPERA CARACTER, 234.
TCL ESPERA TECLA, 114, 234.
TCL ESTADO JOYSTICK, 118, 236.
TCL EXAMINA ESC, 122, 239.
TCL EXAMINA TECLA, 117.
TCL GENERA RUPTURA, 121-122, 238.
TCL IMPONE REPETICION, 120, 237.
TCL IMPONE RETARDO, 120, 238.
TCL INHIBE RUPTURAS, 121, 238.
TCL INICIALIZACION, 113, 233.
TCL LEE CARACTER, 114, 118, 234.
TCL LEE TECLA, 114, 234.
TCL OBTIENE CONTROL, 119, 237.
TCL OBTIENE MAY, 237.
TCL OBTIENE SHIFT, 119.
TCL PERMITE RUPTURAS, 121, 238.
TCL PONE ESTADO, 236.
TCL PONE EXPANSION, 116, 235.
TCL PONE TRASLADO, 119.
TCL REGRESA CARACTER, 114-115, 234.

TCL REINICIALIZACION, 113, 233.
TCL TECLADO LIBRE, 234.
TCL TOMA ESTADO, 118, 235.
TCL TOMA EXPANSION, 117, 235.
TCL TOMA TRASLADO, 119, 237.
TCL TRASLADO CON CONTROL, 119, 237.
TCL TRASLADO CON MAY, 236.
TCL TRASLADO CON SHIFT, 119.
Temporización lenta, 45, 51.
Temporización rápida, 51.
Tintas, 72.
Tipo ROM, 208.
Tipos de ficheros, 136.
TXT ACTIVA VENTANA, 226.
TXT ACTIVA CURSOR SISTEMA, 87, 224.
TXT ACTIVA CURSOR USUARIO, 87, 223.
TXT ACTIVA VDU, 86, 223.
TXT ACTIVA VENTANA, 91.
TXT ADMISION DE GRAFICOS, 100, 228.
TXT CANJEA CAUCES, 93, 226.
TXT CODIGO CARACTER, 95, 97, 228.
TXT COLOR PAPEL, 90, 225.
TXT COLOR PLUMA, 90, 225.
TXT CURSOR, 224.
TXT CURSOR ACTIVADO, 89, 100, 224.
TXT CURSOR ELIMINADO, 89, 100, 224.
TXT ESCRIBE CARACTER, 96-97, 227-228.
TXT EXAMINA CURSOR, 87, 223.
TXT EXAMINA MATRIZ, 94, 227.
TXT EXAMINA MODO, 91, 226.
TXT EXAMINA PAPEL, 90.
TXT EXAMINA PLUMA, 90.
TXT EXAMINA TABLA CONTROL, 99.
TXT EXAMINA TABLA MATRICES, 94, 227.
TXT EXAMINA VENTANA, 88, 92, 226.
TXT FIJA COLUMNA, 86, 223.
TXT FIJA FILA, 86, 223.
TXT FIJA MODO, 90, 225.
TXT FIJA TABLA DE MATRICES, 94, 227.
TXT IMPONE MATRIZ, 95, 227.
TXT INHIBE CURSOR SISTEMA, 88, 224.
TXT INHIBE CURSOR USUARIO, 87, 224.
TXT INICIALIZA, 38, 85, 223.
TXT INVIERTE COLORES, 90, 225.
TXT LEE CAR, 100, 228.
TXT LEE CARACTER, 100.
TXT LIMPIA VENTANA, 92, 226.
TXT PONE CURSOR, 88, 224.
TXT POSICION VALIDA, 89, 225.
TXT QUITA CURSOR, 88, 224.
TXT REINICIALIZA, 85-86, 223.
TXT SALIDA, 97, 228.
TXT SELECCIONA CAUCE, 68, 93, 226.

Unidad de video para gráficos 103, 229.
Unidad de video para texto, 85, 223.
USING, 170.

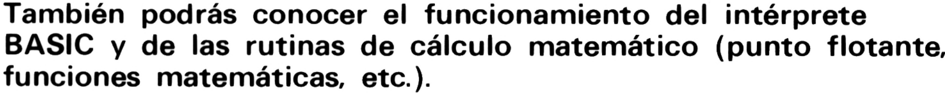
VDU de gráficos, 103.
VDU de texto, 63, 82, 229.
Ventanas, 91.
Video Gate Array, 18.

&, 18.

El sistema operativo de una máquina es el programa que define su comportamiento y características fundamentales. Conocer su estructura de conocimiento es una labor compleja, pero imprescindible, para programar aprovechando al máximo todas las posibilidades del ordenador.

El sistema operativo de los AMSTRAD CPC464 y CPC6128 tiene más de 250 puntos de entrada a rutinas y funciones. **AMSTRAD. Desensamblado de la ROM y mapa de memoria** contiene la descripción de la estructura del sistema operativo, además de la explicación detallada de todas las rutinas, su posición, sus parámetros, su modo de acceso, etc.

Con la información contenida en **AMSTRAD. Desensamblado de la ROM y mapa de memoria** podrás manejar en profundidad la pantalla (Texto, Gráficos, Color, Ventanas, etc.), el sonido, el teclado, las unidades de disco y cassette, etc.



También podrás conocer el funcionamiento del intérprete BASIC y de las rutinas de cálculo matemático (punto flotante, funciones matemáticas, etc.).

¡Introdúcete en el sistema operativo y haz realidad todo ese mundo de posibilidades que se te ofrecen!

ANAYA
MULTIMEDIA


AMSTRAD
Desensamblado de la ROM
Don Thomasson



AMSTRAD
CPC
OCR
MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA
ACME
https://acpc.me/